IPSO FACTO

ISSUE #6
JUNE, 1978

(A publication of the Association of Computer Experimenters)

TABLE OF CONTENTS

		PAGE
1	ACE Executive and Meeting Schedule, 78-79	2
2	Editor's Remarks	3
3	A Software Standard for Kansas City Standard Tapes	5
4	Letters to the Editor	5
5	Abstracts from Dr. Dobb's Journal	10
6	A Game of LIFE for the 1802	11
7	Items for Sale	23
8	A CMOS 16x32 Video Display	23
9	Build a 1K Video RAM	29
10	Reply from NETRONICS LTD President	35
11	Letters of Contact	36
12	An Extra Page of ELF Memory	36
13	Instant Editor	38
14	Interfacing a BAUDOT Teletype to an 1802	38
15	A Mouse-Trap Game for "Pixie Graphics"	47
16	The 1802 Programmer's Notebook	50
17	Erratta	55
18	An 1802 RAM System	57
19	Combination Lock and Door Chime	59
20	A Different Cassette I/O Routine	61
21	Winning Ticket Draw Program	65
22	Minutes of ACE Meetings	68
23	T-Shirts and Logo Contest	72
24	Renewal Membership and Membership Forms	73

Editor : Tom Crawford
Invaluable Assistants: Wayne Bowdish, Diane York, and all contributors to this issue

Information furnished by IPSO FACTO is believed to be accurate and reliable. However, no responsibility is assumed by IPSO FACTO or the Association of Computer Experimenters for it's use; nor for any infringements of patents or other rights of third parties which may result from it's use.

All Newsletter correspondence should be sent to:
Tom Crawford,
50 Brentwood Drive,
Stoney Creek, Ontario,
CANADA L8G 2W8

ACE EXECUTIVE COMMITTEE

In accordance with the Constitution, the 1978-79 Executive Committee approved at the Annual General Meeting is:

President	KEN BEVIS	220 Cherry Post Drive, Missisauga, Ontario, L5A 1H9 (277-2495)
Past President	TOM CRAWFORD	50 Brentwood Drive, Stoney Creek, Ontario, L8G 2W8 (662-3603)
Secretary/ Treasurer	GEORGE YORK	60 Chester Road, Stoney Creek, Ontario, L8E 1Y2 (664-5264)
Newsletter Editor	BERNIE MURPHY	102 McCrany Street, Oakville, Ontario, L6H 1H6 (845-1630)
Program Co-ordinator	BERT DEKAT	P.O. Box 137, Lynden, Ontario LOR 1TO (647-3931)
Training Co-ordinator	NORM CUNNINGHAM	129 Barons Aveune North, Hamilton, Ontario, L8H 5A6 (549-3477)
Hardware Co-ordinator	FRED FEAVER	105 Townsend Avenue, Burlington, Ontario, L7T 1Y8 (637-2513)
Membership Co-ordinator	WAYNE BOWDISH (Temporary)	149 East 33rd Street, Hamilton, Ontario, L8V 3T5 (388-7116)
Newsletter Publishing Committee	DENNIS MILDON	44 Wildewood Avenue, Hamilton, Ontario, L8T 1X3 (385-0798)
	JOHN HANSON	955 Harvey Place, Burlington, Ontario, L7T 3E9 (637-1076)

There are still some positions to be filled eg. software and membership co-ordinators. Any volunteers for these positions or to help with any other activity should contact Ken Bevis.

CLUB MEETINGS

There will be no club meetings during the summer months. Unless notice to the contrary, the following meeting schedule will commence in the fall--

Date	Tutorial/Meeting	Tutorial
Sept 12	7:00/8:00	
Sept 26		7:00
Oct 10	7:00/8:00	
Oct 24		7:00
Nov 14	7:00/8:00	
Nov 28		7:00
Dec 12	7:00/8:00	

Also- unless notice to the contrary, all meetings will be held in the Stelco Wilcox St. Auditorium.

Looking back over the founding and development of the Association of Computer Experimenters and IPSO FACTO, I must say I am amazed and pleased at what we have managed to do in less than one year. We have gone from a group of a half-dozen with brand-new computer kits and some far-out ideas, to a group in excess of 500, from literally all parts of the world, with a large number of well-developed home computer systems. We are linked together by one of the best computer club newsletters around, supported by a wonderful group of contributors, which is easily the envy of any other newsletter, amateur or professional.

There are reportedly more than 400 computer clubs on the North American continent. We may not have the largest one (yet), but I feel we provide a most important service to our members: the communication of ideas and facts. It is the intention of the new ACE Executive to continue and improve this service in the next year. I am pleased with our new Executive; I know most of them personnally, and I am sure they will all do an excellent job.

REPRINT POLICY

We have had several people inquire about IPSO FACTO's reprint policy, so here is what we have come up with (our thanks to DDJ):

"Articles herein that are copyrighted by individual authors or otherwise explicitly marked as having restricted reproduction rights may not be reprinted or copied without permission from the authors. All other articles may be reprinted for any non-commercial purpose, provided a credit line is included. The credit line should credit the author, and should indicate that the material was reprinted from IPSO FACTO, a publication of the Association of Computer Experimenters.

AUTHOR'S ADDRESS POLICY

To promote communication of questions and ideas relating to particular articles we print, we shall in the future print the mailing address of all authors of articles and letters, unless the author specifically states otherwise. We encourage anyone with questions, or needing more information on an article to contact the author directly. This will reduce the amount of letter re-routing done by the IPSO FACTO Editor.

MEMBERSHIP RENEWALS

Effective June 1st, 1978 renewals are required for all 1977-78 members, in order to continue receiving IPSO FACTO in 1978-79 (6 issues are planned). Note that if we haven't received your renewal by mid-August, 1978, you won't be getting Issue #7, which is planned for the end of August. Elsewhere in this issue you will find a renewal form and instructions. Note that the annual fee for 78-79 is $10.00.

NETRONICS AND THE VIPER

This Editor recently received a letter from Mr. George Meyerle, President of Netronics Research and Development Ltd., which should answer a lot of questions asked by those of you who joined Netronics' Users' Group many months ago, and haven't heard a thing since. Look for Mr. Meyerle's letter in this issue.

Those of you with RCA VIPs will be interested in a newsletter dedicated to the support of the COSMAC VIP. Volume 1, Issue 1 (16 pages) came out in June, 1978, and included a number of articles, such as Tape Read/Write Routines, and a primer on use of the CHIP-8 language. Single copy price is $2.00, annual subscription rate is US $15.00 (ten issues per year). For more information contact the Editor, VIPER, P.O. Box 43, Audubon, PA 19407.

A SOFTWARE STANDARD FOR KANSAS CITY FORMAT TAPES

Bernie Murphy
Oakville, Ont.

Many owners of microprocessors are in the process of getting a cassette tape interface up and running or already have completed this task.

If we wish to exchange programs on tape, a certain set of rules regarding data format must be set.

In order to get some "standard" going, the following format is recommended:

I) At least 15 seconds of mark (2400 HZ) at the beginning of a file.

II) All data bytes are 8 bits plus one start bit, even parity and one or more stop bits.

III) The Simplified format as discussed in the third issue of IPSO FACTO be used but slightly modified.

- file type	1 byte	ASCII character upper case S
- start address high	1 byte	load address high
- start address low	1 byte	load address low
- byte count	2 bytes	16 bit length (hi then lo byte)
- data byte 1	1 byte	8 bits ⎫
.	.	. ⎪
.	.	. ⎬ n = the value of the byte count
.	.	. ⎪
- data byte n	1 byte	8 bits ⎭
- checksum	1 byte	8 bits of checksum formed by adding all bytes excluding file type & checksum byte

If anyone has any other ideas, please contact me or the editor.

REFERENCES

1. IPSO FACTO issue #3 p. 36.
2. BYTE's Audio Cassette Standards Symposium, Byte Magazine, February 1976, p. 72.

LETTERS TO THE EDITOR

Tom:

I am in the process of putting together a newsletter supporting RCA's COSMAC VIP.

RCA's new marketing manager for the VIP assures me that he (and RCA) will be co-opertive and informative and gave me your name.

I'd like to act as a sort of central clearing house for VIP information exchange. I have the full support and co-operation of several VIP user groups, and have permission to reprint selected articles from their newsletters. I'd like permission from the ACE group to reprint from your newsletter, too.

The "sample" issue for which there will be no charge, should be ready to mail by June 1st, and will be mailed to you (now that I have your name) and to anyone else in your group who let me know he/she wants it.

I'm looking forward to seeing your newsletter and to getting acquainted with your group.

Thanks, Terry L. Laudereau, "The Viper", P.O. Box 45, Audubon, PA 19407, USA

(Terry: Thank you for your letter. Our reprint policy is similar to that of Dr. Dobb's; please see the Editor's Remarks section of this issue of IPSO FACTO. T.C.)

Tom:

The COSMAC SOURCE BOOK is a continuing, non-profit newsletter for 1802 users all over. In CSB you will get: 1)Hardware-whats available and where to get it. 2)Software-original programs printed in the updates. And addresses to write and get lots more software you might not have thought existed! 3)People. Addresses of other users groups, and other handy things. Send $1 to THE COSMAC SOURCE BOOK
1227 10th Ave.
San Francisco, CA 94122
You get the book (in a paper binder), which is continuing. Every couple of weeks you will receive a 1-3 page update.

Sincerely, Karl Mosgofian

Dear Mr. Crawford,

We are a small group of individuals in the Los Angeles area who have gotten together to form an Elf Club. We are just getting started and are in need of any help or advice we can get. It has come to my attention that you are with a huge well run club up in Canada. We would most appreciate hearing from you with any suggestions as to how we may correspond and exchange information. I am enclosing a check to cover the cost of your subscription.

Many thanks. Sincerely, Ron Reis

Dear Mr. Crawford,

After talking to Bernie Murphy recently I aquired an interest in your Hamilton Computer Club. We also have started a club in the Oshawa area and go by the handle "Durham Computer Club". We are still quite small, with a membership of 20. But we have only had 2 meetings and I hope the interest will grow in the future. The large percentage of computers owned by club members are 6800's from Motorola. We do not have a news letter yet but that will be our next consideration.

Myself & my friend would like to apply for membership in your club. We may not be able to make many meetings but hope to find interesting reading in your newsletter. Also if you ever find any bargains on periferals and memory could you please let me know. Thank you.
Yours truly, Vic Willis, 1442 Wecker Dr., Oshawa, Ontario.

Dear Tom:

TINY BASIC is supposed to be ready soon (so they say) from Quest. It is to be on two 2708 EPROMS at 8400 and 8800 (hex). Tom Pittman is doing it and RCA is said to be picking up the cost. It appears that Netronics will have it too. I have my order in.

...I certainly agree that you should publish commented assembly language. No doubt. My problem is that I have no way of checking such a program (that is, one I have dis-assembled). To that end I enclose an order for a cross-assembler....It is not fair to expect you to dis-assemble my programs; indeed I would like to do it myself. Then I can send them to you for publication. (Great! T.C.).

...(Regarding the same 1802 article published in both Popular Electronics and Dr. Dobb's...TC) I sent that article to PE on July 5, 1977. On August 18 I sent a letter asking about its status. I heard absolutely nothing from PE so in September I sent the articles to DDJ. Finally on November 21 PE sent me galley proofs! PE never even let me know that they had received my article or inquiry. When I finally heard from PE it was too late to do anything about the situation.

Regards, Mack. (Edward McCormick)

Tom:

Thanks for your rapid reply to my letter. Since receiving your letter I have received the three back issues of IPSO FACTO so there is no need to send me them again. Also, my 2 friends have finally received their back issues. This morning I received IPSO FACTO #5, boy is it fat! It sure takes a long time for the issues to get to the West Coast, though.

Presently, I'm making a paper-tape recorder so I can read Tiny Basic into my COSMAC. I'm really looking forward to getting Tiny up and running. The paper-tape reader will cost about $15 ($11 of that is for the photo transistors). If it works OK, perhaps I'll write it up for the newsletter. Regarding my inquiry on the 1854 UART, my problem can be stated quite simply:

The Uart will not transmit properly unless the TR bit in the control register is reset to Ø. In the RCA data book, they say TR should be set to 1 for transmission. It took me a couple of days to realize that the UART transmitted OK only if TR = 0.

Tom, I would be interested to see if anyone else has played with this chip and possibly had similar difficulties. Perhaps you could write a little blurb on the above problem in an upcoming IPSO FACTO. That's all for now-I look forward to receiving IPSO FACTO #6.

Regards, Clint Steinberg VE7CDE

Dear Sir:

I would like to thank you for the back issues of IPSO FACTO. I did not realize that there were many 1802 users, and your existance has come as a pleasant surprise.

I have two micro's running, a Netronics Elf and an RCA VIP. I had originally purchased the Elf with the intention of expanding it, but the board I received from Netronics had two shorts between memory lines and an intermittent break where the load line passed

through the board. It took a few weeks to sort out the bad keys and find a .015 cap which was missing. It also took them three months to send me the power supply I ordered with it. I have yet to receive their newsletter.

The VIP has been a different story. It took about two hours to assemble and has worked perfectly. I have had no problem expanding memory or finding the IC's for the I/O port.

I have been ordering parts to expand my system and I hope you will find the following of some interest. I have ordered the UT4 from RCA to use along with the tiny BASIC I have. It will plug directly into the socket used by the ROM (566) supplied with the VIP and uses the same address. The tiny BASIC is set up to use the ASCII R/W routines provided in the UT4. I will mount both ROMs and use a DIP sw to select which system I want as a monitor. I have also ordered a 4K memory board from Quest with a TTY interface and S-100 memory interface. I hope this will be easy to kludge into my system. I have also ordered a used KSR 33 for $300 and will let you know how it works.

I have been using a GE - 5105B cassette recorder to record programs from the VIP and the following method seems to work every time.
Record: Volume MAX, Tone lowest position.
Load: Volume 3/5, Tone highest position

Also, I am using a TV-1 I purchased from JADE, which works fine with the VIP, but does not work with the Elf. I have been hooking it to the UHF antenna and tuning to the lowest UHF station I can. This seems to give me a more stable picture and since it is used mostly for games I don't really care how sharp the image is.

I have written a few simple programs in CHIP-8 and adapted the PE music program to the ELF. I am working on a CHIP-8 program which will allow me to use the VIP as a calculator. So far, it will add three decimal numbers and display four. I will include a copy of it when I get at least four functions working. (We're waiting, Richard! T.C.) Thank you. Richard Blessing

Dear Mr. Crawford:

I am having trouble with the CPU (1802CD) on my TEC-1802. The instructions 67, 62, 63, and 1N do not function properly. Has anyone had any similar problems? The other functions work.

I would be interested in corresponding with anyone interested in a low-cost A/D converter. I am thinking of using Exar's 2240 or Intersil's 8240. One example of an A/D converter may be found in Popular Electronics Feb. 1977 in the "Solid State" column.
Sincerely, Chris Airhart, 364 Church St., Stratford, Ontario N5A 2S1

Tom,

Thanks for your prompt response. Your club sounds very active & exciting.

...Please send as many back issues of IPSO FACTO as possible-particularly issue number 5. I have been very interested in S-100 conversion (if possible) and MAG tape storage, and many other subjects covered in your newsletter.

My RCA-1802 system is configured as follows:

Circuit derived from Popular Electronics articles
Hexadecimal pad input
1K memory board (2102)
Q line speaker
Toggle switch input

I am presently building a mother board using Molex right angle P.C. connectors (a la SWTP 6800), a digital I.C. test board (both TTL & CMOS), & 3 I-O Port board.

My entire system including PC boards are entirely homebrew.

I am extremely interested in your 8K memory board and Kansas City standard cassette interface, and TVT video displays. I have been looking for a suitable TVT interface, but have not found one as yet. Please send me as much info on these projects as you can and I will forward additional information on both my software and hardware.

A friend and I are working on a new language (not BASIC). It will include 4 Byte accuracy in all math calculations, 26 variables, and will utilize Reverse Polish notation. At present, it looks like it will require only 1 or 2 K of memory.

Thanks again for your response and I look forward to receiving IPSO FACTO. Gregg Germain

(Thanks for your letter, Gregg. Note the 2 TVT articles in this Issue. Also watch for a third article in the next Issue. (We have to get the interface running yet.) We would greatly appreciate receiving more information on both your hardware and software, Gregg. Perhaps in the form of newsletter articles? Keep in touch. Tom).

Dear Tom,

Thanks for sending me the back issues of IPSO FACTO. They have provided me with several hours of informative reading. I hadn't realized that you were so well organized.

I currently own a "Super Elf" made by Quest Electronics in California. It recently won the Science Fair at my school. I would like to sell it, perhaps to a member of the club. I have enclosed information.

Can you send me a copy of the pinout of the Tektron 1802 Bus? I would like to build a compatible system. (Done. TC)

Is anyone up there considering a 24x80 video display? I am thinking about building one in the near future and would appreciate any ideas. I'm thinking more of a stand-alone unit with integral memory and RS-232 data transfer rather than a system-dependent unit, as I would like to use it on the computer at school as well. (Wow! Lets hear about it! T.C.)

Does anyone have any ideas for a S-100 bus interface? I like the prices on 16k memory boards and would like to use one. A schematic would be appreciated.

Do you have diagrams for a simple RS-232 interface and software to use it? I would also be interested in parallel interfaces and software as I am not very good at assembly language programming.

Keep up the good work!!!

Sincerely, A.D. Barkdale Garbee II, Student/Experimenter, 1601 Clayton Avenue, Lynchburg, Virginia 24503

P.S. I might be able to organize a branch of the club here in Lynchburg. (Any interested persons, please contact him directly. I think its a tremendous idea. T.C.)

ABSTRACTS FROM DR. DOBBS JOURNAL

Paul V. Birke
Burlington, Ont.

DDJ is the definite source for software in the publishing world of personal computing in North America. To quote from DDJ "In the two years of publication DDJ has carried a large variety of interpreters, editors, debuggers, monitors, graphics, games software, floating point routines and software design articles." DDJ also publishes independent evaluations on products produced for the personal computing market. Since DDJ is a subscriber-supported journal, i.e. carries no paid advertising, its evaluations are as unbiased as is possible.

Abstracts of articles in DDJ will appear in IPSO FACTO if they meet either of the following two conditions:

1. The article directly pertains to the RCA COSMAC 1802 uP system.
2. The article is considered by this reviewer to be of general interest to the membership.

1) David M. Allen, "Use a PROM for a Character Generator," Vol. 2, No. 5, p. 17, May 1977.

In a letter David discusses the use of the 1702A PROM as 8-by-8 dot matrix character generator using address lines A0, A1, and A2 for row select while lines A3 through A7 are used for character select. Although limited in speed, he used the AM1702ADC to make a 26 line by 64 column CRT display. As David points out, "the existence of even faster PROM'S guarantees their usefulness as character/graphics generators."

2) Joe Weisbecker, "A Practical, Low-Cost, Home/School uP System," Vol. 2, No. 5, pp 34-44, May 1977.

Joe discusses the approach to a minimum uP system called FRED developed around the COSMAC 1802. It contains 1K RAM, Hex Keyboard, cheap cassette, and user TV set (RF). The philosophy of use is one of education, games and true creative experimentation. Joe estimates today's cost as less than $200.

This lengthy article describes in some detail many aspects of this low cost system concept. Many applications are given, some software of which have already been fully developed for the COSMAC microprocessor.

3) Edward McCormick, "Utilities & music on the COSMAC Elf", No. 19, Vol. 2, Issue 9, pp. 30-33.

Ed's article lists and explains software which reads and writes programs and data to and from a regular cassette recorder. Also given are listings to play music, to interface with a Teletype, to measure frequencies (f max = 5800 Hz) and to measure time intervals. Hardware requires a 2 Mhz crystal plus simple cassette and Teletype interfaces.

Of interest is the fact that the software given implements the "Kansas City" standard for cassette interfacing.

ABSTRACTS FROM DR. DOBB'S JOURNAL (CONT'D)

4) Edward McCormick, "Promable 1K RAM plus 256 byte EPROM plus cassette recorder plus simple transistor cassette interfacing hardware can overcome a limited memory, lack of an OP and volatility of memory". Same Issue as above.

The operating system performs five basic functions:
1. starts program execution at any place in memory.
2. reads from memory.
3. writes to memory.
4. reads from cassette recorder to RAM.
5. writes from RAM to cassette recorder.

Cassette software for interfacing described in previous article (above). Operating system stored in a 1702 PROM with battery back-up.

MACHINE-LANGUAGE GAME OF LIFE PROGRAM FOR COSMAC 1802

by Ben Hutchinson
Lexington, Mass.

Dear Tom,

I appreciated your prompt and friendly response to my inquiry. I am also going to look into Pitman's TINY BASIC and that offered by Infinite, Inc. I assume, however, that these are pretty much stripped down: integer arithmetic, no strings, etc.

...The interest you expressed got me started documenting my LIFE program; subsequently, a few others inquired also. A description is enclosed which should be detailed enough to allow it to be easily adapted to any COSMAC system with 1K of RAM, and 61 video chip, and some sort of keyboard and monitor to allow entry of any data to any address. I don't know your Tektron at all, but what I read of the ELF convinces me it would run on one.

Writing the program was both challenging and fun, and running it even more so. We have spent hours just trying new patterns. The generations come at 2½ second intervals, so its never boring.

I'd like to see the program widely published and enjoyed; I'm pleased at the idea of people having fun with it on small simple COSMAC systems. I hope you can publish it in the club newsletter. Feel free to make lots of copies and pass them out at meetings or any other way.

I sent my dues to your membership man right away. I'm particularly interested in the back issues describing the TVT-6 and keyboard interfaces; haven't yet heard.

Let me know how you and others make out with LIFE, and if you discover or find published any new LIFE-forms not described in the 1970-72 Sci. Am. articles. I'll send to anyone who gets it running a few patterns not in Sci. Am. which my son and I have discovered which are long-lived or otherwise interesting.

I tested the program in its 1K version, with interrupt routine in RAM, on my VIP and it ran fine.

Regards, Ben Hutchinson, 30 Partridge Rd., Lexington, Mass 02173

The game of LIFE was invented by mathematician John Horton Conway in 1970. Though not conceived as a computer game originally,

it quickly became popular as such. Surely it is one of the more delightful and absorbing of computer games, especially with a CRT display. Although it can be and is played with a keyboard/printer terminal, it is essentially graphical in nature rather than verbal or numeric, and really needs none of the usual keyboard interchange ("The Klingons are coming! Enter move"). All these things suggest that it might be ideally suited to a COSMAC-based microcomputer equipped with an 1861 video interface chip and nothing more than a hexidecimal keyboard. This turns out to be not only true, but easier than you might think. The accompanying program uses only 1024 bytes (4 pages) of memory, of which half is used for current and alternate display refresh for the standard 32 X 64 display format. The program takes about 2.5 seconds per generation. It was developed and run on an RCA COSMAC VIP, but can be easily adapted to run on any of the ELF or similar small COSMAC systems so long as they have an-1861, 1K or more of memory, and some way to write arbitrary bytes to selected addresses anywhere in memory.

A complete description of the Game of LIFE may be found in Scientific American for October, 1970, page 120. Important follow-up articles appeared as follows: November, 1970; January, February, March, and April, 1971; November, 1971; and January, 1972. The name derived from an analogy with the growth of colonies of living organisms, such as bacteria. The idea is to start with a simple pattern of live "cells" (in a COSMAC, 1's in memory giving spots of light) and see how it grows when certain very simple genetic growth laws are applied. Every cell in a square array has 8 "neighbours" (counting diagonals). The rules are: Each cell with two or three neighbours survives. Each cell with four or more neighbours dies from overcrowding. Each cell with one neighbour or none dies from isolation. Each empty space with exactly three neighbours has a birth (live cell in next generation). All births and deaths occur simultaneously; together they produce a new pattern for the next "generation". The changes are unusual, sometimes beautiful, always unexpected. Very simple initial patterns can grow to enormous complexity and persist for hundreds of generations.

In the program, two pages of RAM are used for display refresh storage. One stores the current generation's pattern, and is accessed by the interrupt service routine, using R0 as the DMA pointer, as controlled by interrupt and DMA requests from the 1861 video chip. The other page stores the next generation's pattern, as it is built up bit by bit by the program, which of course applies the rules listed above, counting the number of live neighbours for each cell in the current generation. The way the 1861 and 1802 work together allows both display refresh and access to data for testing to go on separately and independently, without of course affecting the display of the current generation. When all cells of the current generation have been tested, and a new generation is ready in the alternate page, the "current" and "new" page addresses are simply swapped between the appropriate R registers, so that the RAM page containing the new pattern begins to refresh the display.

Besides the basic loop which decides and writes the new-generation pattern and periodically swaps display pages, the program contains an initialization routine and a routine to clear the first-used display

page. The only other functions needed are the interrupt-servicing routine for display refresh and some way of conveniently entering 1's in selected locations of the first display page, such as from a hex keyboard. In the VIP, these last two tasks are easily done by routines contained in the ROM operating system. For an ELF or similar machine, all that is needed is to add in RAM the interrupt service code similar to that on the 1861 data sheet or in the PIXIE articles (July-August '77 Popular Electronics), plus some sort of keyboard monitor/operating system such as EHOPS (or even ETOPS) described in Pop. Elec. March 1977, or the ELF-II monitor program, March 1978 Pop. Elec. It will still all fit in 1K, even if the keyboard and interrupt routines are not in ROM.

In order to understand how the heart of the program works (I.E., that part that counts the number of live neighbours which each cell has, and applies the rules of LIFE), refer first to Fig. 1. This is a map of 9 words in the display-refresh page of RAM, with their addresses relative to the center word shown. They are drawn the same way that the 1861 draws their contents on the CRT: in ascending address order, words are painted left to right in 8-word rows, with the first row across the top of the screen. In the description and listing, the word in which bits (cells) are currently being checked is called the "A" word. The "A" word address is stored in RA. The eight neighbour words shown in Fig. 1 are obviously those having one or more neighbour bits of bits in the "A" word, and thus involved in testing that word. These words are referred to by their relative addresses; e.g., "the (A+9) word", not to be confused with the (A+9) address, sometimes called "the address of the (A+9) word".

Figure 2, also drawn the same way the 1861 draws on the CRT with the MSB on the left, shows the designation used for individual bits in each 8-bit word, i.e., the hex value of a word in which only that bit is a "1". This notation is convenient and is used throughout the listing; it corresponds to the hex value of a pointer or mask which picks out the single bit which it designates.

Brief study of Figs. 1 and 2 together with the rules of LIFE given above make clear several key facts on which the structure and operation of the COSMAC program are based:

1) All neighbours of the center 6 bits of the "A" word (02 thru 4) are contained in only 3 RAM words: A, (A-8), and (A+8).
2) In contrast, the 80 and 01 bits (MSB and LSB) clearly require separate and more complex treatment, since in each case three additional RAM words contain at least one neighbour bit, for a total of six words.
3) Testing all the neighbours of the 40 bit in the A word is particularly straightforward: all the neighbour bits are in three words, and all can be easily shifted left into DF where a single BNF instruction tests 0 or 1.
4) Special provisions must be made when the A word is at the edge of the display area, and one or more words containing neighbour bits do not exist on the display page.

These facts are the key to the organization of the program. As shown in the flow chart of Fig. 3, there are three main code segments concerned with the actual testing of neighbours of all 8 bits

in a particular "A" word: one for testing the 80 bit, one for testing the center 6 bits (02 through 40) which uses a loop, and one for testing the 01 bit. Each of these code segments loads the A word and its neighbours from RAM, after testing whether A is an edge word. It then shifts the A, (A-8), and (A+8) words so that the particular bit being tested, with its neighbours, is shifted to occupy the 40 bit position. The shifted words are stored in R6.1, R7.0, and R7.1. For the 80 and 01 bits, single neighbour bits from RAM words to the left and right respectively of the A word in Fig. 1 are brought in via DF as part of this process. Finally, all three code segments call a single subroutine, SUB 40, which applies the rules of LIFE to the 40 bit in R6.1 and its neighbours in R7.0 and R7.1, as described under (3) above. SUB 40 then writes 0 or 1 in the actual bit position of the A word on the "next-generation" display page. It knows where to write by using the "A alternate" word address in RF, and a pointer to the actual tested bit in RE.0. It knows what to write by testing R6.0, which it used to count the number of 1's among the 8 neighbour bits.

The remaining code consists of rather straightforward and obvious controls to increment and reset the A word address after all bits in a word have been tested, and to swap display pages when the pattern for a new generation has been completely written. There is also a simple subroutine which merely performs 5 left shifts (SUB 5). It is used in testing the "01" bit.

The handling of edge words can be done several ways. One published BASIC version of LIFE simply stops the program whenever the rules would call for a live cell outside the displayed area. In this program, zeros are entered in the working registers (R6 and R7) whenever the A word touches an edge, in place of whatever might be in noneexistent neighbour words. This has the effect of killing or distorting growth when a pattern hits an edge, but at least the run continues and one can watch other parts of the pattern develop. Tests are made for top or bottom edge before testing the 80 bit. Such words are labelled by making RC.1 or RD.0 respectively zero; otherwise, these registers are loaded with the address (A-8) and (A+8) respectively. Left-edge words are detected during setup of the 80 bit test, while right-edge words are checked before test of the 01 bit.

With a VIP, operation is as follows. The program is loaded with a C0 (LBR) at location 0027. When the switch is first thrown to RUN, initialization is followed by a jump to the erase routine at 0160, which clears page 6 and calls the operating system. Best next step is to immediately write a C8 (NLBR) in 0027, so that the patterns you are about to enter on page 6 don't get erased when you forget! Then, call the operating system in the normal way (key C held down while switching to RUN) and use the memory write mode to enter the starting pattern somwhere on page 6 (repeat this step as needed). The four center words are 067B, 067C, 0683, and 0684; making a chart helps. Finally, switching to RUN with no keys held and a C8 at 0027 starts the evolution, which stops only with a stable pattern or a switch to RESET. With 1 K, use pages 2 and 3; the listing shows how.

The operation can be made less clumsy in a number of ways, like automatic or key-dependent clearing of the erase flag. However, if

you want this, program it before you play the game, or you'll be having too much fun to bother, like I did! An ELF, with its hardware-switch mode selection and ability to start execution anywhere in memory, may be able to avoid clever or dependent branching entirely by selecting starting addresses for erase, monitor, or run.

The R registers (all of them!) are used as follows:

R0: Initial program counter, then DMA refresh (initialized by interrupt routine to beginning of display page in use)

R1: Interrupt service program counter. For VIP, initialize to 8146 before turning on the 1861 chip with a 69 instruction. For ELF or other, initialize to entry point of interrupt routine.

R2: Stack pointer. Used only by interrupt routine in this program. Initialize to 01FF, or any place with 2 free bytes.

R3: Main program counter (immediate handover from R0)

R4: SUB-40 subroutine program counter

R5: SUB-5 subroutine program counter

R6.0: 1's counter, used by SUB-40 to count 1's among the eight neighbour bits

R6.1: Working storage for the shifted "A" word being tested

R7.0: Working storage for the shifted (A-8) word being tested

R7.1: Working storage for the shifted (A+8) word being tested

R8 Changed by VIP interrupt. Do not use unless using a

R9 different routine which does not increment them

RA: Address of "A" word. INC or DEC for A+1 or A-1 words

RB.1: Display page pointer. VIP interrupt routine loads it into R0.1 before starting DMA display refresh.

RB.0: Used by VIP interrupt routine for scratch pad

RC: Address (A-8) if A is not top-edge word; C.1=00 if top edge

RD: Address (A+8) if A is not bottom-edge; D.0=0 if bottom edge

RE.0: Single bit =1 to point to bit being tested in "A" word. Center-6-bits loop exited when pointer reaches 01 bit.

RF: Address of "A" word on "other" (next-generation) display page

(A-9)	(A-8)	(A-9)
(A-1)	"A" WORD	(A+1)
(A+7)	(A+8)	(A+9)

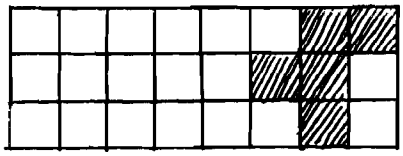
FIG. 1: Array of memory bytes as displayed by 1861, showing RAM addresses relative to center word

80 40 20 10 08 04 02 01

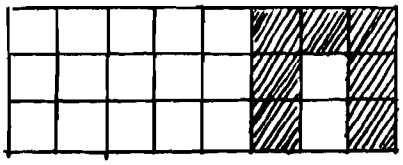
MSB LSB

(A-8)	80	40	20
A	80	40	20
(A+8)	80	40	20

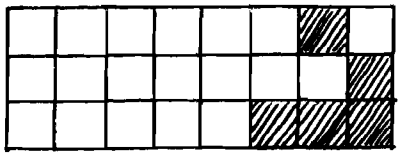
FIG. 2: Designation of individual bits in a byte, and the 8 neighbours of the 40 bit



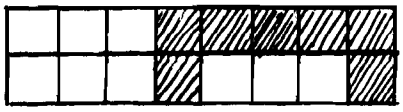
DATA	ADDRESS	
03	066D	
06	0675	The R Pentomino
02	067D	



07	0673	
05	0678	"PI"
05	0683	



02	0673	
01	067B	The Glider
07	0683	



1F	0673	Pulsar CP 48-56-72
11	067B	

Some LIFE Forms and how to enter them in memory.

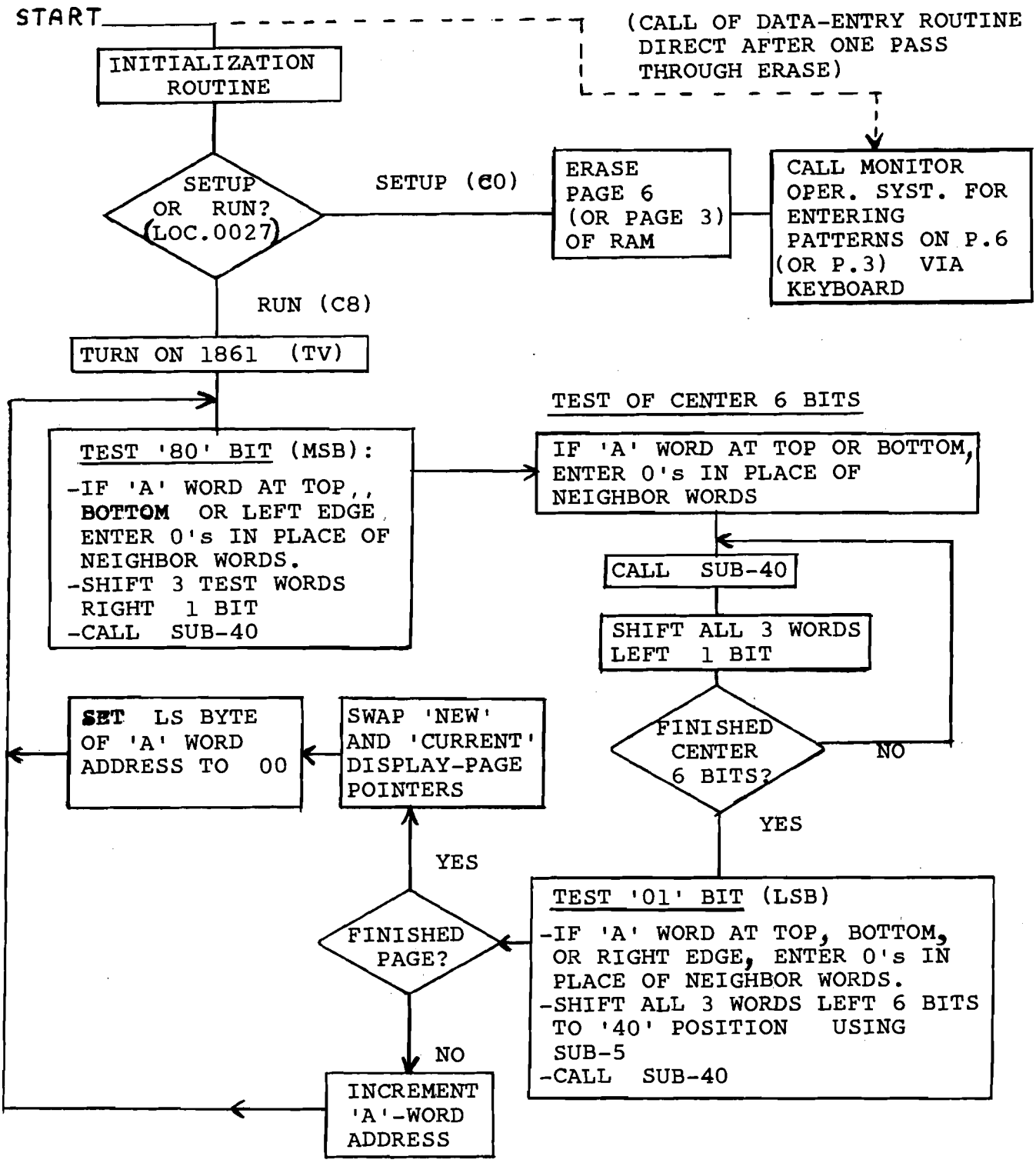

START
INITIALIZATION ROUTINE
(CALL OF DATA-ENTRY ROUTINE DIRECT AFTER ONE PASS THROUGH ERASE)
SETUP OR RUN? (LOC.0027)
SETUP (C0)
ERASE PAGE 6 (OR PAGE 3) OF RAM
CALL MONITOR OPER. SYST. FOR ENTERING PATTERNS ON P.6 (OR P.3) VIA KEYBOARD
RUN (C8)
TURN ON 1861 (TV)
TEST '80' BIT (MSB):
-IF 'A' WORD AT TOP,, BOTTOM OR LEFT EDGE, ENTER 0's IN PLACE OF NEIGHBOR WORDS.
-SHIFT 3 TEST WORDS RIGHT 1 BIT
-CALL SUB-40
TEST OF CENTER 6 BITS
IF 'A' WORD AT TOP OR BOTTOM, ENTER 0's IN PLACE OF NEIGHBOR WORDS
CALL SUB-40
SHIFT ALL 3 WORDS LEFT 1 BIT
FINISHED CENTER 6 BITS?
NO
YES
SET LS BYTE OF 'A' WORD ADDRESS TO 00
SWAP 'NEW' AND 'CURRENT' DISPLAY-PAGE POINTERS
YES
FINISHED PAGE?
TEST '01' BIT (LSB)
-IF 'A' WORD AT TOP, BOTTOM, OR RIGHT EDGE, ENTER 0's IN PLACE OF NEIGHBOR WORDS.
-SHIFT ALL 3 WORDS LEFT 6 BITS TO '40' POSITION USING SUB-5
-CALL SUB-40
NO
INCREMENT 'A'-WORD ADDRESS

FIGURE 3

```
                        ;  GAME OF LIFE  -  MACHINE LANGUAGE PROGRAM
                        ;
                        ;    FOR COSMAC 1802 WITH 1861 VIDEO INTERFACE
                        ;    - WRITTEN BY BEN H. HUTCHINSON, JR. 1978.
                        ;
                        ;       INITIALIZATION
                        ;
0000  90                        GHI RO          ; (RO.1=00)
0001  B3                        PHI R3          ; SET R3=0006 AND MAKE
0002  F8 06                     LDI MAIN        ; IT THE PROGRAM COUNTER
0004  A3                        PLO R3
0005  D3                        SEP R3
0006  F8 81              MAIN:   LDI #81         ; SET R1=ENTRY ADDR. OF DISPLAY-REFRESH
                                                ; INTERRUPT SERVICE ROUTINE: 8146 FOR VIP
0008  B1                        PHI R1          ; ROM OPERATING SYSTEM. FOR ELF OR OTHER,
0009  F8 46                     LDI #46         ; WITH ROUTINE IN RAM PAGE 1, CHANGE
000B  A1                        PLO R1          ; TO 0182 (OR OTHER RAM OR ROM ADDR.)
000C  F8 01                     LDI #01
000E  B2                        PHI R2          ; SET R2 (STACK POINTER) TO 01FF
000F  F8 FF                     LDI #FF         ; (OR ANYWHERE WITH 2 FREE LOCATIONS:
0011  A2                        PLO R2          ; USED ONLY BY INTERRUPT ROUTINE)
0012  92                        GHI R2
0013  B4                        PHI R4          ; SET SUBR. COUNTER TOP HALVES (R4.1,
0014  B5                        PHI R5          ;  R5.1) TO PAGE 1 (GET 01 FROM R2.1)
0015  F8 01                     LDI SUB40
0017  A4                        PLO R4          ; R4 NOW = 0101, SUB-40 ENTRY POINT
0018  F8 51                     LDI SUB5
001A  A5                        PLO R5          ; R5 NOW = 0151, SUB-5 ENTRY POINT
001B  90                        GHI RO          ; (=00)
001C  AA                        PLO RA          ; SET RA.0 AND RF.0=00
001D  AF                        PLO RF
001E  F8 06                     LDI #06         ; SET "A"-WORD ADDR. PAGE IN RA.1, AND
                                                ; DISPLAY - PAGE POINTER IN RB.1, TO 06
0020  BA                        PHI RA          ; (FOR VIP WITH 2K MEMORY).
0021  BB                        PHI RB          ;-> CHANGE LOC.001F TO 02 TO RUN
                                                ;    WITH 1K OF RAM
0022  C4                        NOP             ; *SPARE*
0023  BD                        PHI RD          ; SET RD TO SAME PAGE AS RA AND RB.
0024  F8 07                     LDI #07         ; SET ALTERNATE - PAGE POINTER (RF) TO 07
                                                ; (FOR VIP WITH 2K MEMORY)
0026  BF                        PHI RF          ; -> CHANGE LOC.0025 TO 03 FOR 1K OF RAM
0027  C0 01 60                  LBR ERASE               ; (NLBR)
                                                ; C0 CAUSES BRANCH TO ERASE ROUTINE AT
                                                ;  #0160, CHANGE TO C8 TO RUN
002A  C4                        NOP             ; (3 CYCLES)  *SPARES*
002B  C4                        NOP
002C  C4                        NOP
002D  E2                        SEX R2
002E  69                        INP  1          ; TURN ON 1861 T.V. CHIP (SPECIAL TO VIP:
002F  00                        IDL             ;  ELF OR OTHER MAY TURN ON 1861 DIFERENTLY
                        ;
                        ;       TEST OF "80" BIT (MSB)
                        ;
0030  F8 00             TST80:  LDI     #00
0032  BC                        PHI RC
0033  AD                        PLO RD          ; SET RC.1,RD.0, AND R7 TO DEFAULT STATE
0034  A7                        PLO R7          ;     (ALL ZERO)
0035  B7                        PHI R7
0036  8A                        GLO RA          ; IF LOW BYTE OF "A" ADDR.<08, THEN SKIP
0037  FF 08                     SMI #08         ; BECAUSE A IS AT TOP EDGE (LEAVES RC.1=0)
0039  3B 3F                     BNF TOPO10
003B  AC                        PLO RC          ; SINCE "A" NOT AT TOP EDGE, STORE (A-8) IN RC.
003C  9A                        GHI RA          ; GET DISPLAY PAGE IN USE
003D  D3                        SEP R3          ; (2-CYCLE NOP)  *SPARE*
003E  BC                        PHI RC          ; STORE DISPLAY PAGE RC.1
003F  8A                TOPO10: GLO RA          ; IF LOW BYTE OF "A" ADDR.>=F8, THEN SKIP,
0040  FF F8                     SMI #F8         ; BECAUSE A IS AT BOTTOM EDGE (LEAVES RD.0=0)
0042  33 48                     BGE BOTO10
0044  8A                        GLO RA
0045  FC 08                     ADI #08         ; SINCE "A" NOT AT BOTTOM EDGE, CALCULATE
0047  AD                        PLO RD          ;  (A+8) AND STORE IN RD.
0048  8A                BOTO10: GLO RA          ; IF THE LEAST SIGNIFICANT 3 BITS OF THE
0049  FA 07                     ANI #07         ; "A" WORD ADDR. = 000, THEN BRANCH TO
004B  32 6F                     BZ  LFTEDG      ;  #006F BECAUSE "A" IS AT LEFT EDGE.
004D  2A                        DEC RA          ; SINCE "A" IS NOT AT LEFT EDGE,
004E  4A                        LDA RA          ; LOAD THE (A-1) WORD , AND SHIFT ITS
004F  76                        RSHR                    ;  LSB INTO DF
0050  0A                        LDN RA          ; LOAD THE "A" WORD
0051  76                        RSHR                    ; SHIFT THE NEIGHBOR BIT IN DF TO THE 80 POS.
0052  B6                        PHI R6          ; STORE SHIFTED "A" WORD (TEST BIT NOW AT 40 POS.
0053  9C                        GHI RC          ; IF RC.1=0, THEN SKIP SHIFTING (A-8) WORD,
0054  32 5C                     BZ  TOPO20      ;  BECAUSE "A" IS AT TOP OF PAGE
0056  2C                        DEC RC          ; SINCE "A" NOT AT TOP EDGE, LOAD (A-9) WORD.
0057  4C                        LDA RC
0058  76                        RSHR                    ; SHIFT LSB INTO DF
0059  0C                        LDN RC          ; LOAD (A-8) WORD
005A  76                        RSHR                    ; DF -> 80 (MSB)
005B  A7                        PLO R7          ; STORE SHIFTED (A-8) WORD
005C  8D                TOPO20: GLO RD          ; IF RD.0=0, THEN SKIP SHIFTING (A+8) WORD,
005D  32 68                     BZ  BOTO20      ;  BECAUSE "A" IS AT BOTTOM EDGE
005F  2D                        DEC RD          ; SINCE "A" IS NOT AT BOTTOM EDGE,
```

```
0060  4D              LDA RD      ;  LOAD (A+7) WORD
0061  76              RSHR        ;       SHIFT LSB INTO DF
0062  8D              LDN RD      ; LOAD (A+8) WORD
0063  76              RSHR        ;       DF -> 80 (MSB)
0064  B7              PHI R7      ; STORE SHIFTED (A+8) WORD
0065  D3              SEP R3      ; 2-CYCLE NOP  *SPARE*
0066  D3              SEP R3      ; 2-CYCLE NOP  *SPARE*
0067  D3              SEP R3      ; 2-CYCLE NOP  *SPARE*
0068  F8 80    BOTO20: LDI #80     ; SET BIT POINTER/SHIFT MARKER TO 80
006A  AE              PLO RE
0068  D4              SEP R4      ; CALL SUB-40 TO COUNT NEIGHBOR'S 1'S
                                  ;             AND WRITE IN NEW PAGE
006C  D3              SEP R3      ; 2-CYCLE NOP  *SPARE*
006D  30 80           BR  CENTR6  ; GO ON TO TEST OF CENTER 6 BITS
006F  0A       LFTEDG: LDN RA     ; LOAD "A" WORD (A IS AT LEFT EDGE)
0070  F6              SHR         ; MOVE 80 BIT -> 40  (NEW 80 BIT = 0 )
0071  B6              PHI R6      ; STORE SHIFTED "A" WORD
0072  9C              GHI RC      ; IF RC.1=0, THEN SKIP , BECAUSE A
0073  32 78           BZ  TOP030  ;  IS AT TOP EDGE
0075  0C              LDN RC      ; LOAD (A-8) WORD FROM RAM
0076  F6              SHR         ; MOVE 80 BIT -> 40  (NEW 80 BIT = 0)
0077  A7              PLO R7      ; STORE SHIFTED (A-8) WORD
0078  8D       TOP030: GLO RD     ; IF RD.0=0, THEN SKIP,
0079  32 7E           BZ  BOTO30  ;    BECAUSE A IS AT BOTTOM EDGE
007B  0D              LDN RD      ; LOAD (A+8) WORD
007C  F6              SHR         ; MOVE 80 BIT -> 40  (NEW 80 BIT = 0)
007D  B7              PHI R7      ; STORE SHIFTED (A+8) WORD
007E  30 68    BOTO30: BR  BOTO20  ; GO TO 0068, TO SET BIT POINTER AND
                                  ;  CALL SUB-40
                ;
                ;       TEST OF CENTER 6 BITS
                ;
0080  0A       CENTR6: LDN RA     ; LOAD WORD TO BE TESTED ("A"-WORD)
0081  B6              PHI R6      ;  VIA RA, AND PUT IN R6.1
0082  9C              GHI RC      ; IF RC.1 NOT EQUAL 0 THEN FETCH (A-8) WORD
0083  32 87           BZ  CENO10  ;  VIA RC AND PUT IT IN R7.0, FOR "A" IS
0085  0C              LDN RC      ;   NOT AT TOP EDGE; ELSE SKIP TO 0087,
0086  A7              PLO R7      ;    LEAVING R7.0 = 00
0087  8D       CENO10: GLO RD     ; IF RD.0 NOT EQUAL 0, THEN FETCH (A+8)
0088  32 8C           BZ  CENO20  ;  WORD VIA RD AND PUT IT IN R7.1,
008A  0D              LDN RD      ;   FOR "A" IS NOT AT BOTTOM EDGE
008B  B7              PHI R7      ; ELSE SKIP TO 008C, LEAVING R7.1=00.
008C  F8 40    CENO20: LDI #40     ; SET THE BIT BEING TESTED POINTER TO 40
008E  AE              PLO RE
008F  D4       CENO30: SEP R4      ; CALL SUB-40 (COUNT 1'S AMONG 8 NEIGHBORS
                                  ;  OF BIT IN "40" POSITION OF "A" WORD,
                                  ;   WRITE 0 OR 1 IN NEW GENERATION
                                  ;    I.E., IN ALT. DISPLAY PAGE).
0090  96              GHI R6
0091  FE              SHL
0092  B6              PHI R6
0093  87              GLO R7      ; SHIFT ALL THREE WORDS LEFT ONE PLACE
0094  FE              SHL         ;  (A,(A-8), AND (A+8) WORDS, IN R6.1,
0095  A7              PLO R7      ;   R7.0 & R7.1)
0096  97              GHI R7
0097  FE              SHL
0098  B7              PHI R7
0099  8E              GLO RE
009A  F6              SHR         ; MOVE THE BIT BEING TESTED POINTER
009B  AE              PLO RE      ;  ONE PLACE RIGHT
009C  FB 01           XRI #01     ; UNTIL BIT POINTER=01, LOOP TO 008F TO TEST
                                  ; NEXT BIT FROM THE CENTER 6. THEN, MOVE ON
009E  3A 8F           BNZ CENO30  ; TO "TEST OF 01 BIT"
                ;
                ;       TEST OF "01" BIT (LSB)
                ;
00A0  D3       TST01B: SEP R3     ; (2-CYCLE NOP)  *SPARES*
00A1  D3              SEP R3      ; (2-CYCLE NOP)  *SPARES*
00A2  D3              SEP R3      ; (2-CYCLE NOP)  *SPARES*
00A3  D3              SEP R3      ; (2-CYCLE NOP)  *SPARES*
00A4  D3              SEP R3      ; (2-CYCLE NOP)  *SPARES*
00A5  8A              GLO RA      ; IF THE LEAST-SIG. 3 BITS OF THE "A" ADDR.
00A6  FA 07           ANI #07     ; IN RA.0 ARE ALL 1, THEN SKIP GETTING
00A8  FB 07           XRI #07     ; THE MSB'S OF THE (A+1),(A+9),AND (A-7) WORDS,
00AA  32 CC           BZ  RGT010  ; BECAUSE "A" IS A RIGHT-EDGE WORD.
00AC  1A              INC RA      ; INCREMENT RA , TO ADDR. OF (A+1) WORD
00AD  0A              LDN RA      ; LOAD (A+1) WORD
00AE  7E              RSHL        ;       MSB -> DF
00AF  2A              DEC RA      ; RA = "A" WORD ADDRESS
00B0  0A              LDN RA      ; LOAD "A" WORD
00B1  7E              RSHL        ;  MOVE MSB OF (A+2) WORD INTO LSB OF D
00B2  D5              SEP R5      ; CALL SUB-5  (5-SHL'S)
00B3  B6              PHI R6      ; STORE SHFTD "A" WORD,WITH BIT FROM (A+1) WORD
00B4  9C              GHI RC      ; SKIP GETTING (A-7)&(A-8) WORDS IF RC.1=0,
00B5  32 BF           BZ  TOP040  ;  SHOWING "A" IS TOP EDGE
00B7  1C              INC RC
00B8  0C              LDN RC      ; LOAD (A-7) WORD
00B9  7E              RSHL        ;       MSB -> DF
00BA  2C              DEC RC
00BB  0C              LDN RC      ; LOAD (A-8) WORD
00BC  7E              RSHL        ;       DF -> LSB
00BD  D5              SEP R5      ; CALL SUB-5
00BE  A7              PLO R7      ; STORE SHIFTED (A-8) WORD, WITH BIT
                                  ;  FROM (A+7) WORD
```

```
00BF  8D        TOP040: GLO RD      ; IF RD.0=0, SKIP LOAD OF (A+8) AND (A+9)
00C0  32 DF             BZ  RGTEDG  ;  WORDS BECAUSE "A" WORD IS AT BOTTOM EDGE
00C2  1D                INC RD
00C3  0D                LDN RD      ; LOAD (A+9) WORD
00C4  7E                RSHL        ;       ; PUT MSB -> DF
00C5  2D                DEC RD
00C6  0D                LDN RD      ; LOAD (A+8) WORD
00C7  7E                RSHL        ;       ; PUT DF -> LSB
00C8  D5                SEP R5      ; CALL SUB-5 (5-SHL)
00C9  B7                PHI R7      ; STORE SHIFTED (A+8) WORD,WITH BIT FROM
                                    ;  FROM (A+9) WORD
00CA  30 DF             BR  RGTEDG  ; BRANCH AROUND CODE FOR RIGHT-EDGE-WORD
                                    ; SHIFT OF LSB.
00CC  0A        RGT010: LDN RA      ; LOAD "A" WORD ; BEGIN SHIFT OF 01 BIT OF
                                    ;                       RIGHT-EDGE-WORD.
00CD  FE                SHL         ; TO MAKE 6 TOTAL SHIFTS; MOVE 01 BIT TO 40.
00CE  D5                SEP R5      ; CALL SUB-5  (5 LEFT SHIFTS)
00CF  B6                PHI R6
00D0  9C                GHI RC      ; IF RC.1=0, THEN SKIP LOAD OF (A-8) WORD
00D1  32 D7             BZ  TOP050  ;  BECAUSE A-WORD IS AT TOP EDGE (LEAVES R7.0=0)
00D3  0C                LDN RC      ; FETCH (A-8) WORD
00D4  FE                SHL         ; 6 LEFT SHIFTS; 01 BIT TO 40
00D5  D5                SEP R5
00D6  A7                PLO R7      ; STORE SHIFTED (A-8) WORD IN R7.0
00D7  8D        TOP050: GLO RD      ; IF RD.1=0, THEN SKIP LOAD OF (A+8) WORD,
00D8  32 DF             BZ  RGTEDG  ; BECAUSE A-WORD IS AT BOTTOM EDGE (LEAVES R7.1=0)
00DA  0D                LDN RD      ; LOAD (A+8) WORD
00DB  FE                SHL         ; 6 LEFT SHIFTS ; 01 BIT TO 40
00DC  D5                SEP R5
00DD  B7                PHI R7      ; STORE SHIFTED (A+8) WORD
00DE  D3                SEP R3      ; 2-CYCLE NOP  *SPARE*
00DF  D4        RGTEDG: SEP R4      ; CALL SUB-40 (COUNT 1'S AMONG 8 NEIGHBORS
                                    ;  OF BIT IN 40 POSITION, WRITE OOR1 IN NEW
                                    ;                       GENERATION
                ;
                ;       ADDRESS INCREMENTING AND
                ;       PAGE ALTERNATION CONTROL.
                ;
00E0  D3                SEP R3      ; 2-CYCLE NOP  *SPARE*
00E1  D3                SEP R3      ; 2-CYCLE NOP  *SPARE*
00E2  D3                SEP R3      ; 2-CYCLE NOP  *SPARE*
00E3  D3                SEP R3      ; 2-CYCLE NOP  *SPARE*
00E4  8A                GLO RA      ; IF A.0=FF, LAST WORD ON CURRENT DISPLAY
00E5  FB FF             XRI #FF     ; PAGE HAS BEEN TESTED, SO GO TO PAGE ALT.
00E7  22 ED             BZ  ALT010  ; ELSE INCREMENT THE ADDR.ES, AND LOOP.
00E9  1A                INC RA      ; RA NOW HAS ADDR. OF NEW WORD IN CURRENT
                                    ;                       GENERATION
00EA  1F                INC RF      ; RF NOW HAS ADDR. OF NEW WORD IN NEXT
                                    ;                       GENERATION
00EB  30 30             BR  TST80B  ; LOOP TO "TEST OF 80 BIT", START TESTING
                                    ; BITS IN NEW WORD.
00ED  9F        ALT010: GHI RF      ; PAGE ALTERNATION; GET NEW PAGE NO. FROM RF
00EE  BA                PHI RA      ; SET A,(A+8) WORD ADDR. TO NEW PAGE
00EF  BD                PHI RD
00F0  98                GHI RB      ; GET "OLD" PAGE ADDR., NOW ALTERNATE PAGE
00F1  BF                PHI RF      ; SET RF.1 TO ALTERNATE PAGE (NEXT GEN.)
00F2  9A                GHI RA      ; GET NEW DISP. PAGE ADDRESS
00F3  BB                PHI RB      ; SET DISPLAY-PAGE POINTER TO NEW ADDR.
00F4  F8 00             LDI #00
00F6  AA                PLO RA      ; SET A.0,F.0, AND C.1 TO 00.
00F7  AF                PLO RF
00F8  BC                PHI RC
00F9  30 30             BR  TST80B  ; LOOP TO "TEST OF 80 BIT", START TESTING
                                    ; BITS IN FIRST WORD OF NEW PAGE.
00FB  D3                SEP R3      ; 2-CYCLE NOP  *SPARE*
00FC  D3                SEP R3      ; 2-CYCLE NOP  *SPARE*
00FD  D3                SEP R3      ; 2-CYCLE NOP  *SPARE*
00FE  D3                SEP R3      ; 2-CYCLE NOP  *SPARE*
00FF  D3                SEP R3      ; 2-CYCLE NOP  *SPARE*
                ;
                ;       SUBROUTINE "SUB-40": TESTS ALL 8 NEIGHBORS
                ;       OF THE BIT IN THE 40 POSITION OF THE
                ;       SHIFTED "A" WORD (STORED IN R6.1),
                ;       COUNTS THE 1'S, (COUNT IN R6.0), AND WRITES
                ;       0 OR 1 IN THE CORRECT BIT OF THE CORRECT WORD
                ;       IN THE NEXT GENERATION PAGE, ACCORDING TO "LIFE"
                ;   RULES. SHIFTED NEIGHBOR WORDS (A+8) AND (A-8) ARE
                ;       FETCHED FROM R7.1 AND R7.0 FOR THE TEST.
                ;
0100  D3        S40RTN: SEP R3      ; RETURN TO MAIN PROG.
0101  F8 00     SUB40:  LDI #00     ; (ENTRY POINT)
0103  A6                PLO R6      ; RESET 1'S COUNTER (R6.0) TO 00
0104  96                GHI R6      ; FETCH SHIFTED "A" WORD
0105  FA A0             ANI #A0     ; MASK OUT ALL BUT "NEIGHBOR" BITS, 80 AND 20.
0107  FE                SHL         ; SHIFT "80" BIT INTO DF
0108  3B 0B             BNF S40L10
010A  16                INC R6      ; INCREMENT 1'S COUNT IF "80" BIT
                                    ;  WAS 1 ELSE SKIP
010B  FE        S40L10: SHL         ; SHIFT "20" BIT INTO DF
010C  FE                SHL
010D  3B 10             BNF S40L20
010F  16                INC R6      ; INC. COUNT IF "20" BIT WAS 1, ELSE SKIP.
```

```
0110  97        S40L20: GHI R7       ; FETCH SHIFTED (A+8) WORD
0111  FE                SHL          ; "80" BIT -> DF
0112  3B 15             BNF S40L30
0114  16                INC R6       ; INC. COUNT IF "80" BIT WAS 1, ELSE SKIP.
0115  FE        S40L30: SHL          ; SHIFT, TEST, COUNT IF 1, "40" BIT OF (A+8)
0116  3B 19             BNF S40L40
0118  16                INC R6
0119  FE        S40L40: SHL          ; "20" BIT -> DF
011A  3B 1D             BNF S40L50   ; TEST,
011C  16                INC R6       ; COUNT IF 1
011D  87        S40L50: GLO R7       ; LOAD SHIFTED (A-8) WORD
011E  FE                SHL          ; SHIFT, TEST, COUNT IF 1 , THE THREE
011F  3B 22             BNF S40L60   ;  "NEIGHBOR" BITS IN THE (A-8) WORD
0121  16                INC R6
0122  FE        S40L60: SHL
0123  3B 26             BNF S40L70
0125  16                INC R6
0126  FE        S40L70: SHL
0127  3B 2A             BNF S40L80
0129  16                INC R6
012A  D4        S40L80: SEP R4       ; 2-CYCLE NOP  *SPARE*
012B  D4                SEP R4       ; 2-CYCLE NOP  *SPARE*
012C  D4                SEP R4       ; 2-CYCLE NOP  *SPARE*
012D  D4                SEP R4       ; 2-CYCLE NOP  *SPARE*
012E  D4                SEP R4       ; 2-CYCLE NOP  *SPARE*
012F  D4                SEP R4       ; 2-CYCLE NOP  *SPARE*
0130  96                GHI R6
0131  FA 40             ANI #40      ; GET SHIFTED "A" WORD, MASK OUT ALL BUT "40"
                                     ;  BIT, I.E., BIT WHOSE FATE IS TO BE DECIDED
0133  32 3A             BZ  S40L90   ; IF BIT=0, SKIP TEST FOR 2 NEIGHBORS
0135  86                GLO R6
0136  FB 02             XRI #02      ; SINCE "40" BIT=1, GET 1'S COUNT AND
                                     ; BRANCH TO "WRITE-1" ROUTINE IF
0138  32 4A             BZ  WRITE1   ; 1'S COUNT=2 (2 NEIGHBORS: SURVIVAL)
013A  86        S40L90: GLO R6       ; GET 1'S COUNT
013B  FB 03             XRI #03      ; IF # OF 1'S (NEIGHBORS)=3, BRANCH TO
013D  32 4A             BZ  WRITE1   ; "WRITE-1" ROUTINE (SURVIVAL OR BIRTH);
                                     ; ELSE CONTINUE TO "WRITE-0" ROUTINE:
013F  8E                GLO RE       ; LOAD BIT POINTER INTO D (1 AT BIT POS.)
0140  FB FF             XRI #FF      ; COMPLEMENT BIT POINTER (0 AT BIT POS.)
0142  EF                SEX RF       ; "AND" THE COMPLEMENTED BIT POINTER
0143  F2                AND          ; WITH WORD AT LOC.A IN NEW GENERATION PAGE.
0144  5F                STR RF       ; WRITING "0", AND STORE IN LOC.A, "NEW" PAGE.
0145  30 00             BR  S40RTN   ; BRANCH TO SUB-RETURN POINT
0147  D4                SEP R4       ; 2-CYCLE NOP  *SPARE*
0148  D4                SEP R4       ; 2-CYCLE NOP  *SPARE*
0149  D4                SEP R4       ; 2-CYCLE NOP  *SPARE*
014A  8E        WRITE1: GLO RE       ; "WRITE-1" ROUTINE
014B  EF                SEX RF       ; LOAD BIT POINTER
014C  F1                OR           ; "OR" WITH WORD AT LOC.A IN "NEW GENERATION"
014D  5F                STR RF       ; PAGE, WRITING "1", AND STORE IN LOC.A, "NEW"PAGE
014E  30 00             BR  S40RTN   ; BRANCH TO SUB-RETURN POINT
                        ;
                        ;      "SUB-5" SUBROUTINE : SHIFTS D LEFT 5 TIMES
                        ;
0150  D3        S5RTN:  SEP R3       ; RETURN TO MAIN
0151  FE        SUB5:   SHL          ; FIVE LEFT SHIFTS TO MOVE A BIT FROM
0152  FE                SHL          ; 02 POSITION TO 40 POSITION IN WORD.
0153  FE                SHL
0154  FE                SHL
0155  FE                SHL
0156  30 50             BR  S5RTN    ; BRANCH TO SUB-RETURN POINT
0158  =  00 08          .BLOCK  8    ; SPARE LOCATIONS 158 TO 15F
                        ;
                        ;   ROUTINES TO CLEAR DISPLAY-REFRESH PAGE AND
                        ;   CALL OPERATING SYSTEM TO ALLOW ENTRY OF INITIAL
                        ;   "LIFE FORMS". (VIP OPERATING SYSTEM CALL SHOWN)
                        ;
0160  F8 FF     ERASE:  LDI #FF      ; SET RA.0=FF (RA.1 HAS BEEN SET EARLIER TO
0162  AA                PLO RA       ;  NUMBER OF INITIAL DISPLAY PAGE TO BE CLEARED)
0163  EA                SEX RA
0164  F8 00     CLEAR:  LDI #00      ; STORE 00 IN LOCATION M(R(A)), AND DECREMENT RA
0166  73                STXD
0167  8A                GLO RA       ; IF RA.0 NOT EQUAL FF, PAGE NOT COMPLETELY CLEAR;
0168  FB FF             XRI #FF      ; LOOP TO 0164 AND CONTINUE CLEARING.
016A  3A 64             BNZ CLEAR
016C  1A                INC RA       ; INCREMENT RA BACK TO (X)00 FROM (X-1)FF.
016D  F8 07             LDI #07
                                     ; INITIALIZE R-REGISTERS AS REQUIRED
016F  B1                PHI R1       ; SET R1=07FF
                                     ; CALL OPERATING SYSTEM SO THAT
0170  F8 FF             LDI #FF      ; INITIAL "LIFE-FORM" PATTERN DATA
                                     ; CAN BE ENTERED.
0172  A1                PLO R1       ; --> THE CODE LISTED AFTER LOC.016D
0173  F8 80             LDI #80      ; CALLS THE ROM OPERATING SYSTEM
                                     ; OF A COSMAC VIP WITH 2K MEMORY.
0175  B2                PHI R2       ; SET R2=8028
```

```
                                    ; (FOR BIGGER VIP'S, SET R1 TO TOP OF
0176  F8 28            LDI #28       ; TOP RAM PAGE). FOR OTHER COSMAC
                                    ; SYSTEMS (ELF,ELF-II,SUPER-ELF,
0178  A2                PLO R2       ; TEKTRON,ETC.), CODE BEGINNING AT
                                    ; 016D SHOULD BE REPLACED BY CODE (OR
                                    ; BY A BRANCH TO CODE) WHICH ALLOWS
                                    ; ENTRY OF ADDR. AND DATA FROM A
                                    ; KEYBOARD, AND WRITES THE DATA IN
                                    ; THE SPECIFIED ADDRESS ON THE
                                    ; INITIAL DISPLAY PAGE.
0179  D2                SEP R2       ; BR. TO VIP ROM OPERATING SYSTEM AT 8028

017A  =  00 06           .BLOCK  6   ; SPARE LOCATIONS 17A TO 17F
                ;
                ;     OPTIONAL DISPLAY REFRESH INTERRUPT SERVICE
                ;   ROUTINE (IF NOT USING VIP ROM ROUTINE)
                ;
                ;
                ;     (NOTE: TO USE THIS ROUTINE, CHANGE INITIALIZATION
                ;          ROUTINE AS FOLLOWS: LOC. 0007=01, LOC. 000A=82
                ;
0180  42        INTRTN: LDA R2
0181  70                RET
0182  C4        INTRPT: NOP
0183  22                DEC R2
0184  78                SAV
0185  22                DEC R2
0186  52                STR R2
0187  9B                GHI RB        ; GET NUMBER OF DISPLAY PAGE IN USE FROM RB.1
0188  B0                PHI R0        ; LOAD DISPLAY PAGE NUMBER INTO R0, DMA POINTER
0189  F8 00             LDI #00
018A  A0                PLO R0        ;   NOTE: WITH THE EXCEPTION OF LINES
018C  C4                NOP           ;         0087 AND 0088, THIS IS
018D  C4                NOP           ;         ESSENTIALLY THE 32 X 64 ROUTINE
018E  E2        REFRSH: SEX R2        ;         ON THE 1861 DATA SHEET,
018F  80                GLO R0        ;         WHICH SEE FOR COMMENTS.
0190  E2                SEX R2        ;         THE CHANGE ALLOWS THIS
0191  20                DEC R0        ;         ROUTINE TO BE IN ROM
0192  A0                PLO R0        ;         YET DISPLAY ANY PAGE
0193  E2                SEX R2
0194  20                DEC R0
0195  A0                PLO R0
0196  E2                SEX R2
0197  20                DEC R0
0198  A0                PLO R0
0199  3C 8E             BNI REFRSH
019B  30 80             BR  INTRTN

                        .END
```

```
                                RC1802-V01      GOLIFE.RCA      15-JUN-78  32:11:20      PAGE  9
USER SYMBOL TABLE

ALT010 00ED   BOT010 0048   BOT020 0068   BOT030 007E   CEN010 0087   CEN020 008C   CEN030 008F   CENTR6 0080
CLEAR  0164   ERASE  0160   INTRPT 0182   INTRTN 0180   LFTEDG 006F   MAIN   0006   REFRSH 018E   RGT010 00CC
RGTEDG 00DF   S40L10 010B   S40L20 0110   S40L30 0115   S40L40 0119   S40L50 011D   S40L60 0122   S40L70 0126
S40L80 012A   S40L90 013A   S40RTN 0100   S5RTN  0150   SUB40  0101   SUB5   0151   TOP010 003F   TOP020 005C
TOP030 0078   TOP040 00BF   TOP050 00D7   TST01B 00A0   TST80B 0030   WRITE1 014A

        PROGRAM SIZE = 019D
          0 ERRORS DETECTED
```

ITEMS FOR SALE

1. For Sale: One CDP1802CD based "Super Elf" by Quest Electronics. Six digit address/data display, hex keyboard with monitor in ROM, battery memory saver, 256 bytes of RAM, 1861 video controller with support chips, and a limited expansion bus. Supplied with manual. $200.
Write to: A.D. Barksdale Garbee II, 1601 Clayton Avenue, Lynchburg, Virginia, 24503, USA

2. For Sale: Model 19 teletype table; consisting of model 15 printer and punch tape unit (5 level), punch tape reader (5 level) power supply,distribution panel auto start up control and diagrams. Price $125.00 or best offer. Includes two new ribbons, three rolls of paper and punch tape paper.
Eight level paper tape punch. In process of overhaul needs new driver and interface. Asking $40.00. Complete with diagrams.
Solid disk unit fixed heads; clock and sector tracks 10 data tracks 20 sectors per track complete with electronics and diagrams. As is $15.00.
Two power supplies: one 4v @30A will adjust to +5v @20A and +12v @5A in working order asking $20.00 each.
IBM card reader like new with diagram. Asking $140.00.
Eight level paper tape reader like new with diagrams. Asking $120.00.
Call Stephen (416) 545-4060

3. For Sale: Netronics ELF II (as described in Popular Electronics) with video graphics, 256 byte memory, 1802 CPU, hex keyboard and display, along with, four wire wrap 86 pin edge card connectors, video modulator, power supply, amplifier and speaker on the Q line, coaxial cable, and ELFII proto board. ALL ASSEMBLED and TESTED. Also ELFII Expansion Board (One input port, one output port, monitor, RS232 and 20 ma serial interface, cassette i/o interface, and N-line decode) Unassembled. Total cost new $255. First money order, or bank draft for US $180 takes it. Alan Burnes, M.I.U. Fairfield, Iowa, USA 52556.

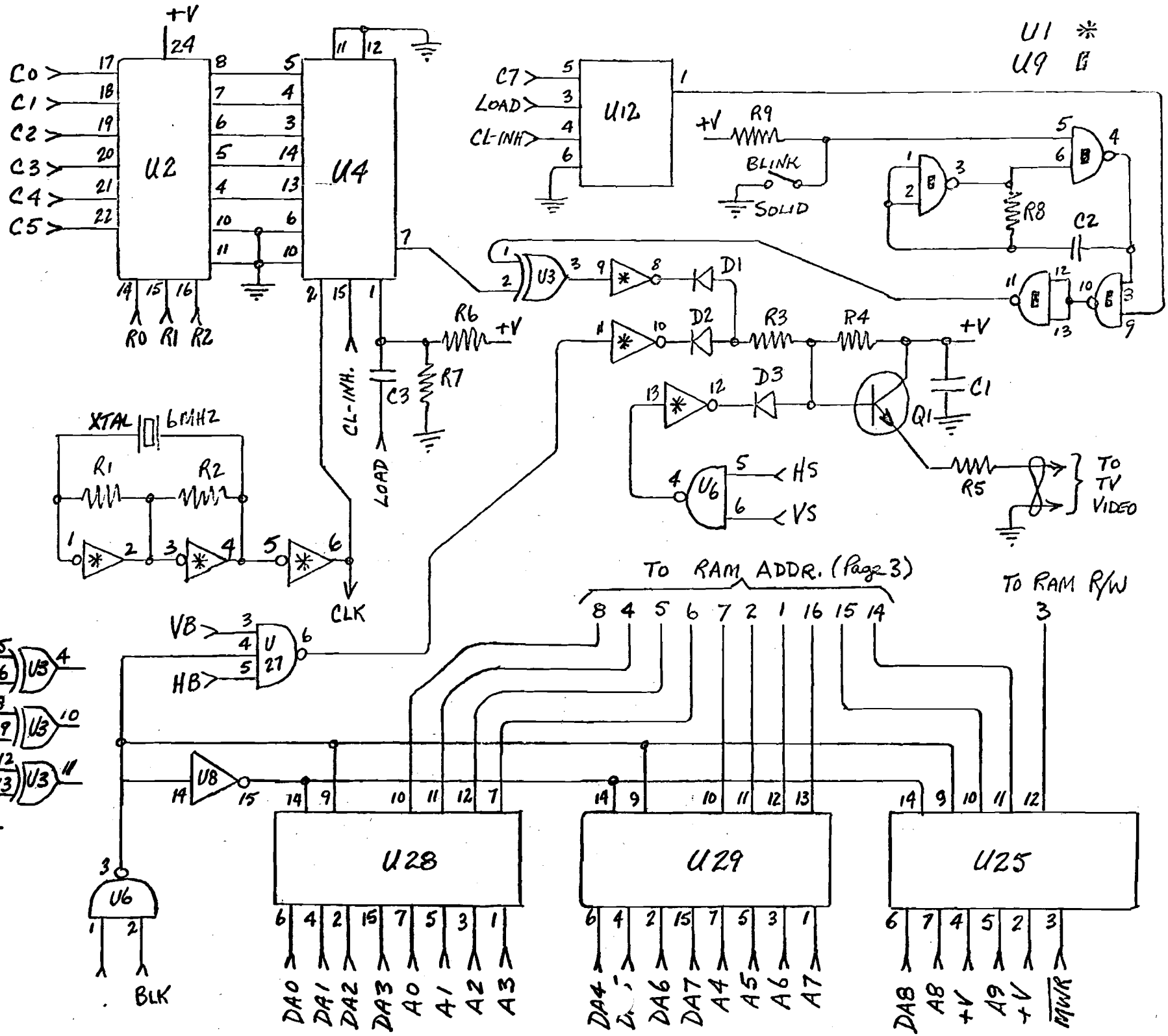
A CMOS 16X32 VIDEO DISPLAY

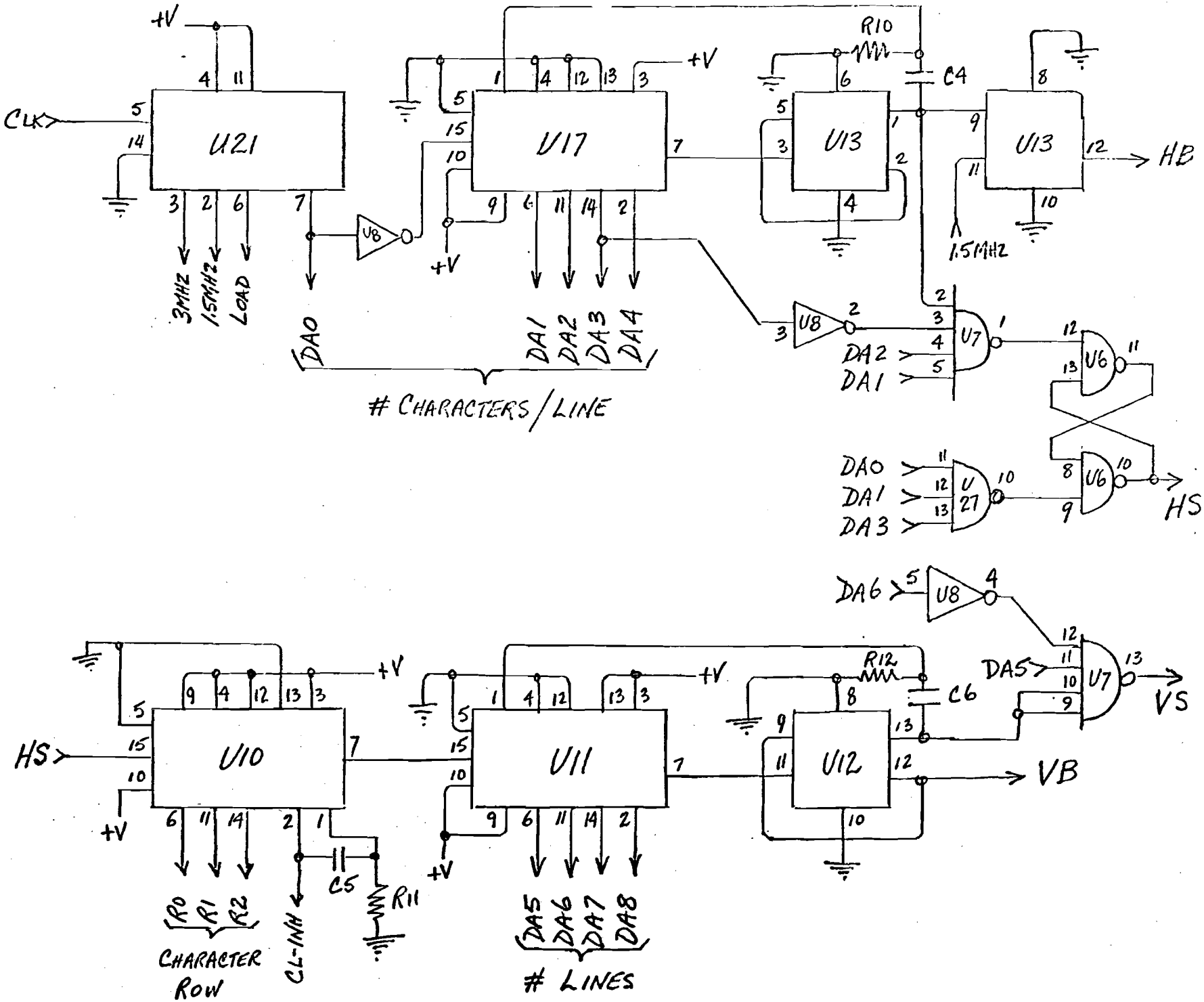
by George Millar (VancouverB(
and Harley Shanko (Van Nuys,C/

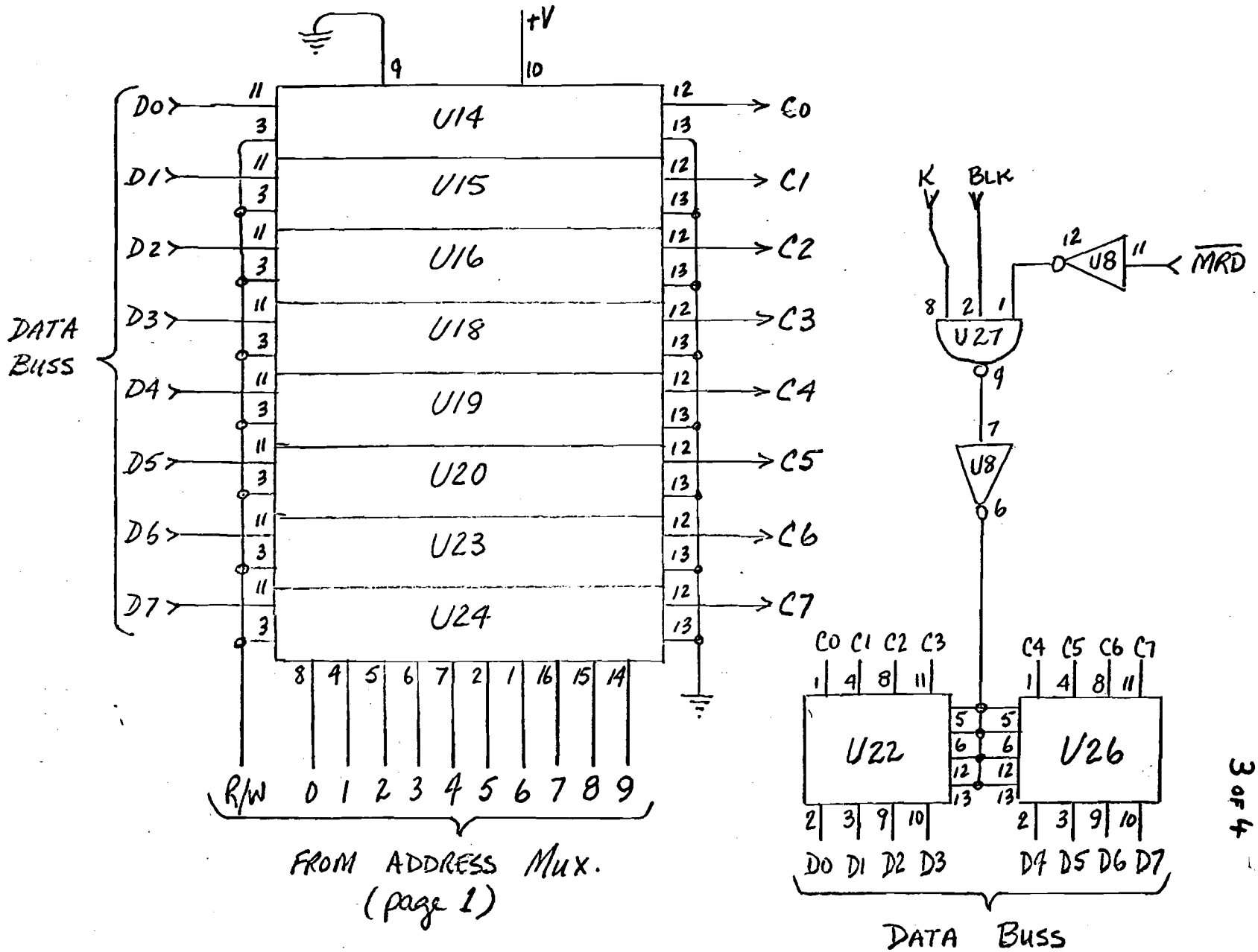
Tom,

I received a letter from Harley Shanko and he said to go ahead and submit the display to IPSO FACTO, he wouldn't mind. He has added that his display is an adaption of the "Suding" version, with numerous modification, the main one being that it is software driven and addressed as a block of memory rather than an output port. Also he has used CMOS where ever possible.

I've enclosed the video card layout, circuit diagrams showing the exact gates used and parts list of my video card as I built it. The circuit is an adaption of Harley's video display modified to suit my 1802 system. I've added the blink or solid cursor circuit to my display. The card layout is of the back side or reverse component side. I didn't include the component side layout of the card as the only thing etched on it is the card name and the other edge connector contacts. Anyone constructing a P.C. card will have to use their own


U1 *
U9 B
+V
C0
C1
C2
C3
C4
C5
U2
U4
R0
R1
R2
C7
LOAD
CL-INH
U12
R9
BLINK
SOLID
R8
C2
U3
D1
D2
R3
R4
R6
R7
C3
CL-INH.
LOAD
XTAL
6MHZ
R1
R2
CLK
D3
C1
Q1
U6
HS
VS
R5
TO TV VIDEO
TO RAM ADDR. (Page 3)
TO RAM R/W
VB
HB
U27
U8
U28
U29
U25
BLK
DA0
DA1
DA2
DA3
A0
A1
A2
A3
DA4
DA5
DA6
DA7
A4
A5
A6
A7
DA8
A8
+V
A9
+V
MWR


+V
CLK
U21
3MHZ
1.5MHZ
LOAD
U8
U17
DA0
DA1
DA2
DA3
DA4
CHARACTERS/LINE
R10
C4
U13
U13
1.5MHZ
HB
U8
U7
DA2
DA1
U6
U6
DA0
DA1
DA3
U 27
HS
DA6
U8
DA5
U7
VS
HS
U10
U11
U12
R12
C6
VB
R11
C5
CL-INH
R0
R1
R2
CHARACTER ROW
DA5
DA6
DA7
DA8
LINES


+V
9
10
U14
U15
U16
U18
U19
U20
U23
U24
D0
D1
D2
D3
D4
D5
D6
D7
DATA BUSS
C0
C1
C2
C3
C4
C5
C6
C7
8 4 5 6 7 2 1 16 15 14
R/W 0 1 2 3 4 5 6 7 8 9
FROM ADDRESS MUX.
(page 1)
K
BLK
U8
MRD
U27
U8
C0 C1 C2 C3
C4 C5 C6 C7
U22
U26
D0 D1 D2 D3
D4 D5 D6 D7
DATA BUSS

PARTS LIST

U1 - 74LS04
U2 - R032513
U3 - 4030
U4 - 74165
U5 -
U6 - 4011
U7 - 4012
U8 - 4049

U9 - 4011
U10 - 4029
U11 - 4029
U12 - 4013
U13 - 4013
U14 - 21L02-1
U15 - 21L02-1

U16 - 21L02-1
U17 - 4029
U18 - 21L02-1
U19 - 21L02-1
U20 - 21L02-1
U21 - 74193
U22 - 4016

U23 - 21L02-1
U24 - 21L02-1
U25 - 4019
U26 - 4016
U27 - 4023
U28 - 4019
U29 - 4019

C1 - .1 ufd.
C2 - .1 ufd.
C3 - 100 pf.
C4 - 110 pf.
C5 - 110 pf.
C6 - 110 pf.

R1 - 1K Ω
R2 - 1K Ω
R3 - 220 Ω
R4 - 470 Ω
R5 - 47 Ω
R6 - 1K Ω

R7 - 1.3K Ω
R8 - 2 Meg Ω
R9 - 22K Ω
R10 - 10K Ω
R11 - 10K Ω
R12 - 10K Ω

NOTE: U5 IS AN UNUSED SOCKET. ALL I.C. SOCKETS ARE WIRE WRAP EXCEPT U1.

D1 - IN4148
D2 - IN4148
D3 - IN4148
D4 - IN4148

XTAL - 6MHZ CRYSTAL
Q1 - 2N2222

+V	GRD.	CONNECTIONS AS FOLLOWS
#14	#7	U1, U3, U6, U7, U9, U12, U13, U22, U26, U27
#16	#8	U4, U8, U10, U11, U17, U21, U25, U28, U29
#10	#9	U14, U15, U18, U19, U20, U23, U24
#24	#10	U2

VIDEO CARD

edge connector size and contact assignment. The display is wirewrapped except for U1 which is a standard solder tail socket. The wiring diagrams are straight forward and easy to follow. I don't think anyone should have problems assembling the video display card. If you require more info please write.....Sincerely, George Millar, P.O. Box 1412, Parksville, B.C.

BUILD A 1K VIDEO RAM

R. Alan Parker Laurel, Md.

Introduction

The circuit described here is a memory mapped video display. The display contains 1K of memory. By a memory mapped display I mean that each character on the video screen is physically represented by one byte in the computer's address space. To the CPU the display circuit simply looks like ordinary memory, i.e. the CPU can read and write into any location. Hardware independent of the CPU scans this 1K block of memory and displays it on the CRT. Since the ASCII characters only use 7 of the 8 bits in a byte, the 8th bit is used to cause the character to blink. If the leftmost bit in a byte is 1, then the character stored in the other 7 bits will blink on the display. The display has the format of 12 lines of 32 characters. At a given time, only about half of the 1K memory is being displayed. Which half is displayed depends upon the setting of a control line. I call these pages, and the control line is the page select line. On my 1802 system the page select line is connected to the Q output. It could just as well be connected to an output port, or not used at all if you only want one page capability. The output of this display is standard NTSC video. If you need RF for an unmodified TV there are several of these available on the market. I would recommend using video, however, as the required modification is not very complex. There have been several articles in BYTE and KILOBAUD showing how this is done. Finally, the memory in this display can be used as ordinary memory if the display function is not needed.

How does this differ from TVT-6?

The circuit for the TVT-6 is much simpler and cheaper than what I am offering. The difference is that TVT-6 is perhaps half software, and my display is all hardware. I do not claim that my way is better; that is a trade off that each user must make for himself. This circuit is always operating and it includes its own memory. To use it, a program need only store into the area set aside for it.

The Circuit

Timing Generator: The most complex part of the display is the timing generator. It must produce a series of addresses to cause a scan through memory at a certain rate. Pulses are also required by the character generator circuit. Synch pulses required by the television monitor are also required, but their generation is simplified by the use of an LSI TV sych generator chip. The 6.13635 Mhz crystal can be ordered from International Crystal, in Oklahoma City. I paid $5 for it, but that was nearly two years ago, so it may be up somewhat. It is important that the inverter IC used as part of the crystal oscillator be CMOS (either a 74C04 or a 4049) or proper oscillation

will not occur. The labeled lines connect to the other drawings, but not all of the lines are used. The 74193 counters are quite hungry power eaters, and if you can afford the higher price 74LS193 or 74C193 would be worthwhile.

Memory: The memory is simple. It's just 1K of RAM made by using the 2102 IC. The data in lines of all 2102s connect to the main memory bus. You should buffer them with 4050s (not shown) to prevent excess bus loading. The address and the data out lines connect to selectors, the next section.

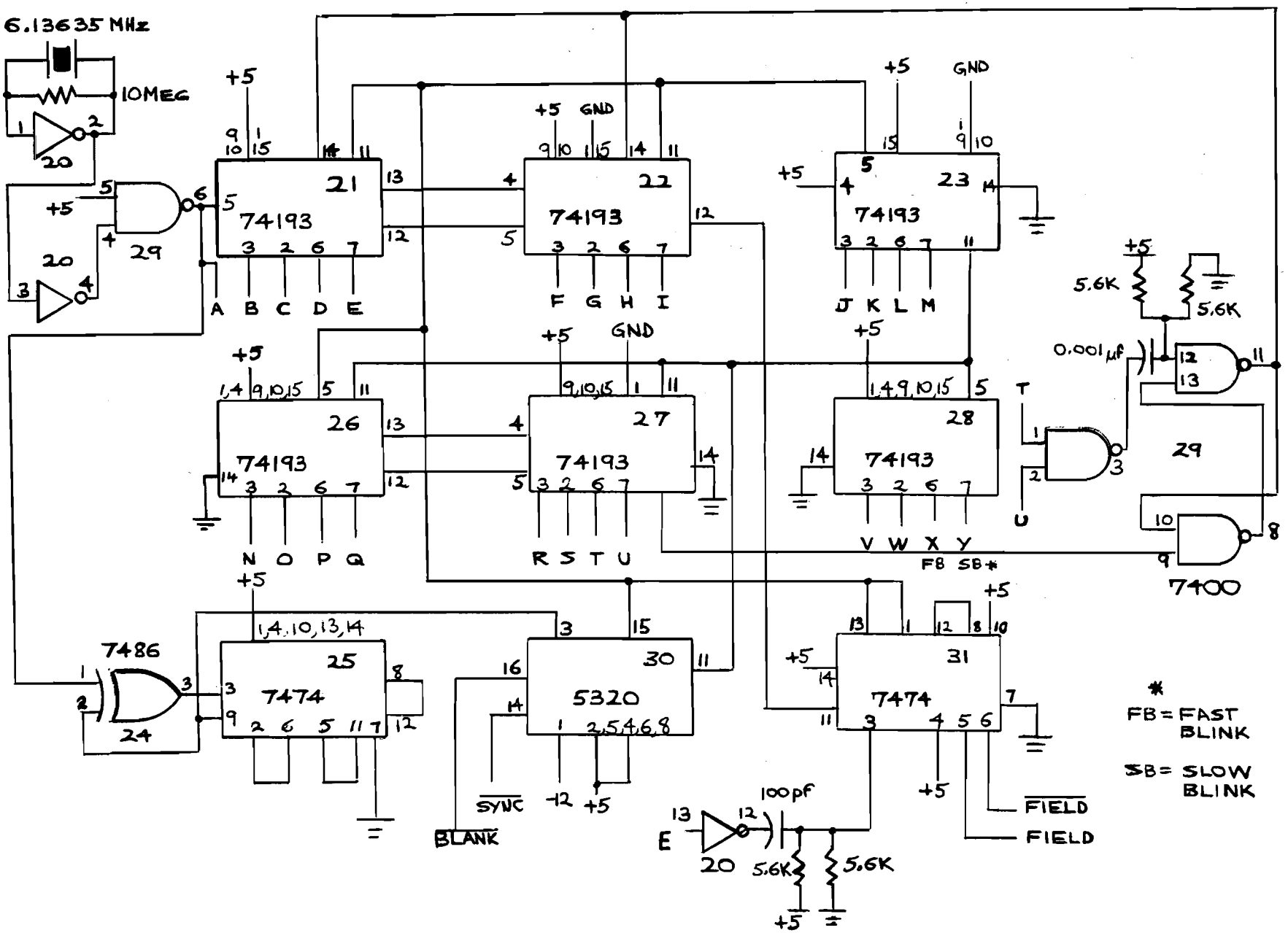
Address Selection and Data Gating: Normally the addresses produced by the timing circuit are applied to the memories to scan them. When the CPU wants to access the display memory it must present its address to the memories and gate the data out onto the bus. The 8122 data selectors are used to switch the address lines of the memories from the display timing circuit to the CPU. You can find the data sheet for them in the National Semiconductor Data (TTL) Book. A single line called BE (bank enable) operates this switch. If BE is low then the memories are addressed by the CPU. If BE is high then the display has control of the memories. Two 4016 quad switches are used to gate the memory data out onto the bus. Tri-state buffers could be used here if you happen to have some. Note how the 1802 MWR and MREAD signals connect to IC17 (8122). The R/W line on pin 12 of IC17 goes to the R/W pin of the 2102s. There are CMOS equivalences to the 8122, such as 4052.

Character Generator Logic: The final section of the display is the character generator circuit. The heart of this is the 2513 character generator ROM. I currently use an upper cases only versio but you can replace this with any type you need. There are some with 128 symbols; upper and lower case and the Greek symbols. These fancy generators cost more. The inputs to the character generator are connected to the outputs of the memory chips. The output is then strobed into the 74165 shift register, and clocked out bit by bit. If bit 7 is 1 then the 7474 at the top of the drawing is set and the blink signal turns the path of the bits on and off (about 1hz). Note the connections for Field, Blank, A, and D refer to the timing circuit. Signal D and the components on it are somewhat critical. Those resisters, while they look useless, are quite important. The transistor can be just about any general purpose signal transistor. The connection of the M line and the NAND gate should be removed if your character generator has more than 7 rows of output. In this case, M is the 4th bit of row information. Do as I have shown if you use a 7 row device such as the 2513.

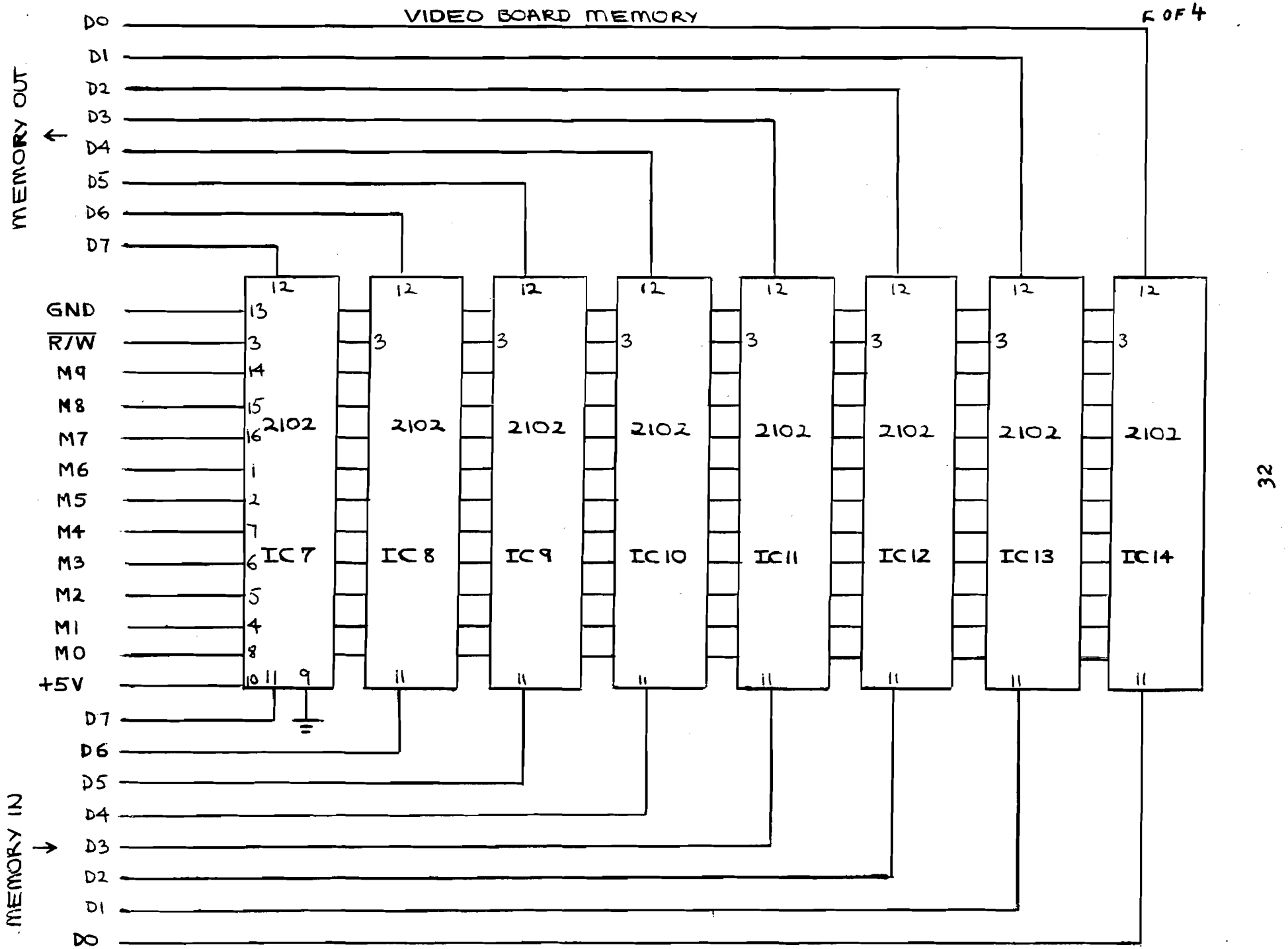
Software - Since the scanning and character generation is done by hardware the software required to use the display is fairly simple. If you want a really simple display for a given program all you have to do is to store the characters in succeeding locations, starting with the address of the upper leftmost corner. When you run off the screen you have two choices; clear the screen and start over, or switch to the other page before wiping things out.

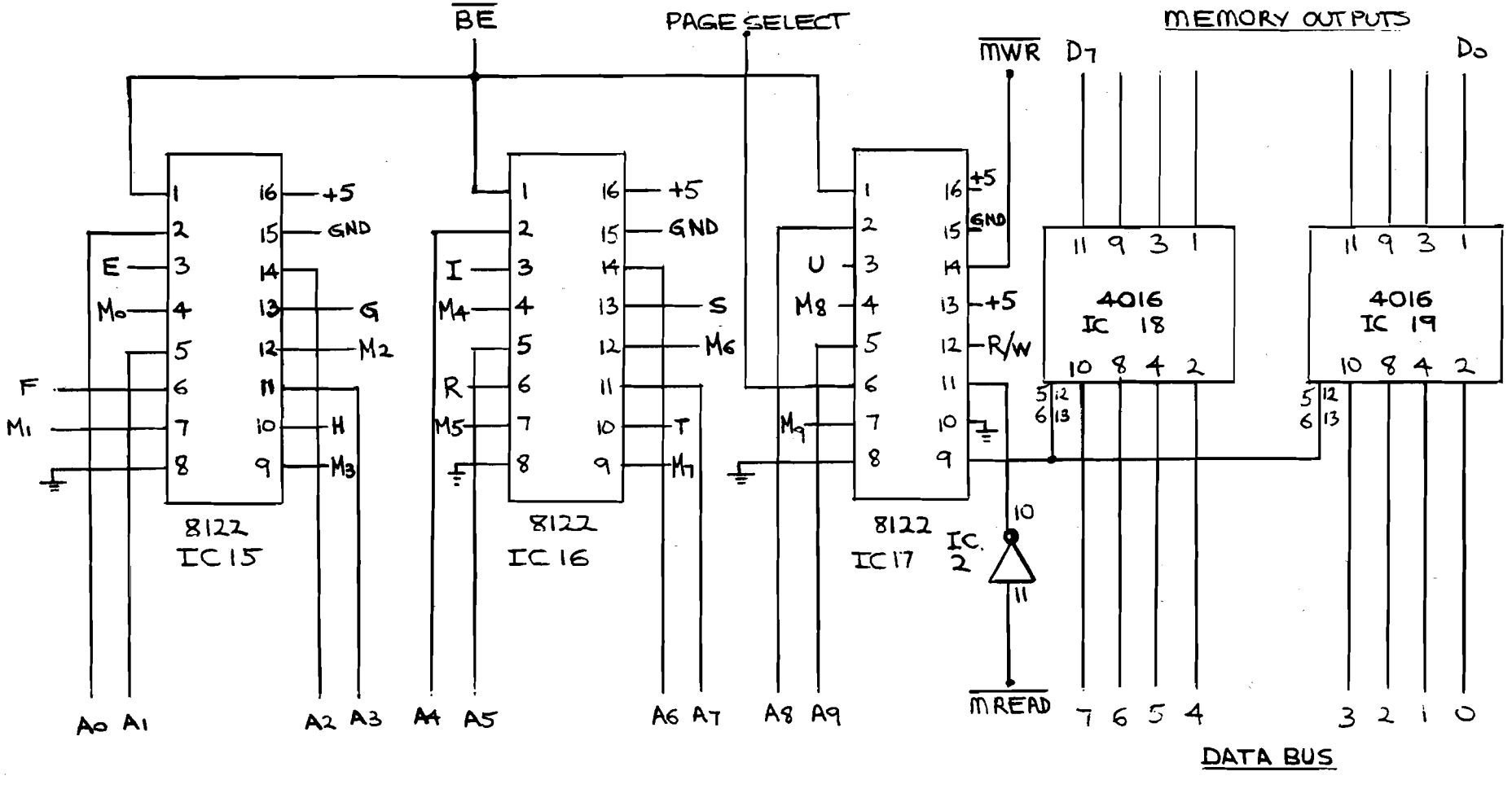
I prefer to start on the bottom line of the display and scroll upward line by line. A sample of the scroll routine is shown. This routine is taken from my I/O system for Tiny BASIC (which I have hac

VIDEO BOARD TIMING

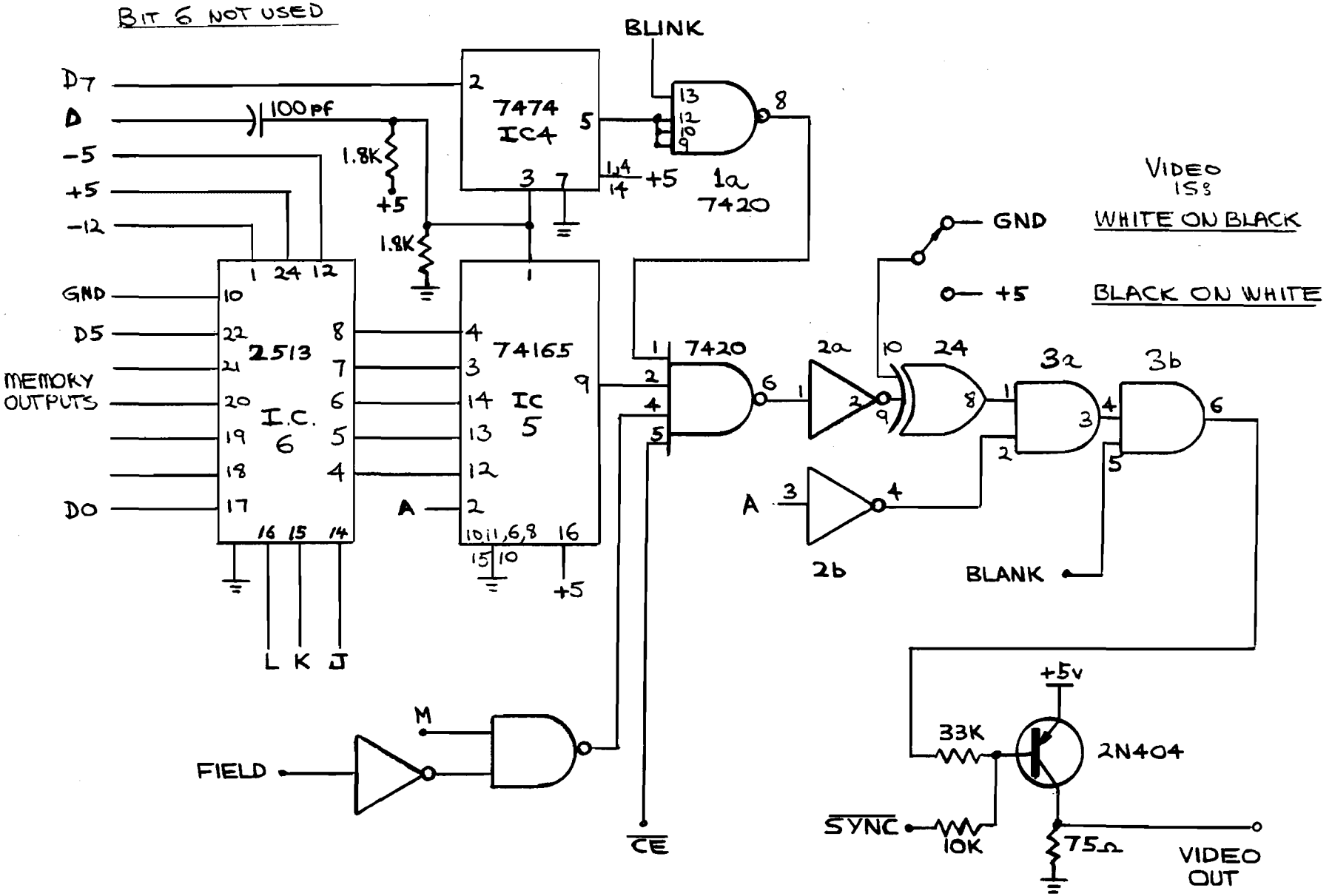


VIDEO BOARD MEMORY




BE
PAGE SELECT
MWR
D7
D0
MEMORY OUTPUTS
+5
GND
E
M0
G
M2
F
M1
H
M3
8122
IC 15
I
M4
S
M6
R
M5
T
M7
8122
IC 16
U
M8
R/W
M9
8122
IC 17
IC. 2
4016
IC 18
4016
IC 19
MREAD
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9
7 6 5 4 3 2 1 0
DATA BUS

VIDEO BOARD CHARACTER GENERATOR LOGIC



working since June 77). The ideal situation would be to place the display routines for scroll and other nice functions in PROM, so that all programs could use them without having to actually include them. I am doing this now.

When the CPU is accessing the display memory the display output is blanked. Under normal conditions this occurs so fast that it is not noticeable on the screen. If, however, the CPU makes many accesses in a short period of time, say more than thousands per second, the display will suffer. This is not a major problem, and the user will quickly learn its limits.

REPLY FROM NETRONICS LTD PRESIDENT

Dear Tom:

In response to your last issue pointing out that we have been delinquent with our Elf II Users Club, please be advised that we are in the process of correcting that situation.

Our first newsletter will contain numerous programs as well as a list of programs to follow. As a consolation we are also offering, at no charge to the Users Club members, a complimentary copy of our new short course on programming 1802 based microprocessors. This course is written in an easy to understand language, and is designed to lead the neophyte through all 91 program instructions giving an illustration and example of how each is used.

Enclosed please find copies of our instruction material for the Elf II, our Giant Board and a 4K memory board. We would appreciate receiving copies of your IPSO FACTO, and if possible the back issues so that we may join with your effort with respect to the 1802.

Thank you in advance for your consideration.

Very truly yours, George M. Meyerle, President
Netronics Research & Development Limited
333 Litchfield Road (Rte. 202)
New Milford, Connecticut, 06776

LETTERS OF CONTACT

Dear Sirs:

Having received your letter, I am more than eager to join the Association of Computer Experimenters. Enclosed is $5.00 to cover the cost of membership & back issues of your newsletter. In response to your request for information on my "computer system", I am making an Elf computer as per plans in Popular Electronics magazine. Having no previous experience in even kit building, I decided to use point to point soldering as recommended in the P.E. article. Looking back, I would have either bought the Elf kit, or used double sided PC board, because of the 1K memory I added on using plans in the P.E. articles. I was forced to use LEDs for readout because I could not locate the necessary HP part. Also, I am including the IC for keyboard (hexadecimal). My system is not up and running yet, for I have not finished soldering, and I have not bought the crystal, and finally, because I cannot find a CD4058BE to run the LED display. I am planning to use a modified P.E. monitor and my uses will be mainly games, mathematical programs (eg. probability, randomness, etc.), and a few relays for controlling apliances. In the past, I have concentrated on software, because I was introduced to computers in school, where I met a Wang 2200. But I find that I am more interested in electronics and solid state experimenting. I plan to add a cassette tape recorder to store programs, and eventually add a TV display. One of my main restrictions is money, since I am a high school student using my own money to build my Elf.

I am looking forward to receiving your newsletter, and would appreciate it if you would let me know if there are any members of your club in my immediate area (Erin Mills).

Sincerely, Paul Brophy, 4131 Saronton Court, Missisauga, Ontario L5L 1Y9

Tom:

I have built a COSMAC Elf and would like to keep in touch with others who have built the COSMAC Elf. Thank you.

Robert Schroder, 23 Ave. D, Holbrook, NY, 11741 USA

AN EXTRA PAGE OF ELF MEMORY

by William P. Webb
San Rafael, CA.

The Elf microcomputer was introduced by Joseph Weisbecker in the August 1976 issue of Popular Electronics magazine. By following his plans, a compact 1802 controlled microcomputer can be produced. It has 256 bytes of memory located in 2-2101 (256 x 4) RAM's.

A second article in the following issue of the same magazine described how this memory could be made permanent by adding a 9v battery and replacing the 2-2101 RAM's with 2-CMOS RAM's. For this purpose, the pin compatible 5101 RAM's are excellent replacements.

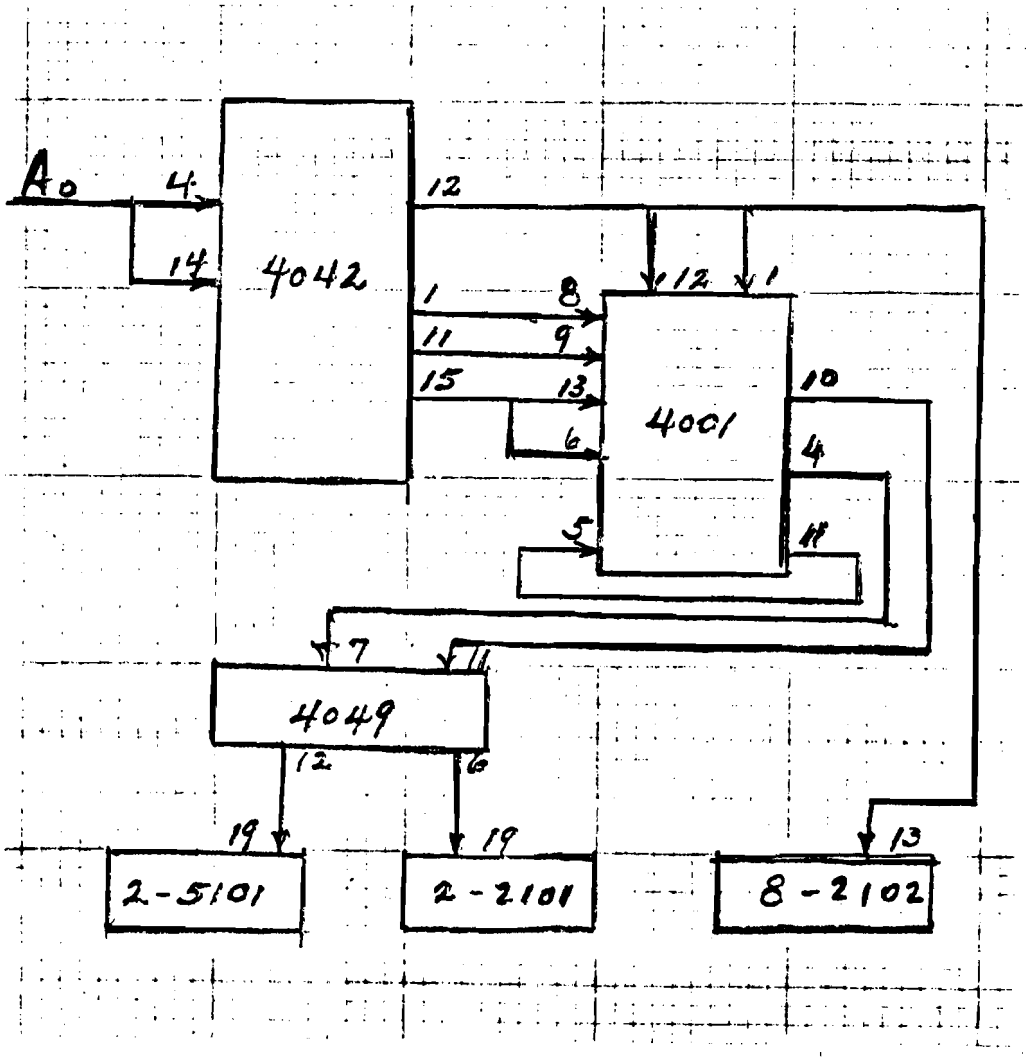
Then a third article in the March 1977 issue of P.E. described how to build a 1K byte addition to the memory, using 8-2102 (1K x 1) RAM's. With this addition, the CMOS RAM memory keeps the address of 0000 - 00FF, and the address of the 1K memory becomes 0400 - 07FF. A 4042 IC is used to select the memory page.

After installing the battery and 1K memory systems, there remains unused: 2-2101 RAM's, 3 of 4 NOR gates on the 4001 IC, and 1 input/output set on the 4042 IC. The remainder of this article describes how to use these left overs, along with an inverter (4049) to add another page of memory to the Elf.

AN EXTRA PAGE OF ELF MEMORY (CONT'D)

Before starting, it should be noted that Fig. 1, on page 65 of the March 1977 P.E. has the pin assignments for A10 and $\overline{A10}$ reversed. The first step of the conversion is to disconnect pin 14 of the 4042 from ground; then disconnect pin 11 of the 4042 from pin 19 of the CMOS RAM's. The remainder of the wiring shown in Figs. 1 and 2 on page 65 of the March 1977 P.E. remains unchanged. Next the 4042, 4001, and 4049 are wired as shown in the diagram below. For the most part only the new wires are shown. Of course, in addition to the pin 19 connection shown, the 2-2101's have to be connected to the address bus, the data bus, and the rest of the control lines as shown in Fig. 4 page 35 of the August issue of P.E.

The result of these modifications is another 256 bytes of memory for the price of one inverter! This memory has the address of 0100 - 01FF.



INSTANT EDITOR

by S. Takahashi
(Montreal Quebec)

Dear Mr. Crawford,

...Here is my little "INSTANT EDITOR" which is only 15 bytes long so can over-write your program and when finished you re-enter original 15 byte, so this Instant Editor disappears.

ADDRESS	OP CODE	USE	
00	F8 04	i)	GOTO the address YYXX which is 000F to FFFF.
02	A1		Press RESET, then RUN and LOAD.
03	D1		Start loading, or verifying with M on, from
04	F8 YY		the address YYXX (Without using Input Switch!
06	B0		for TEC-1802 Users)
07	F8 XX	ii)	RUN from the address YYXX
09	A0		Press RESET, then RUN and "I".
0A	3F 0A		Program starts from the address YYXX when you
0C	37 0C		release the switch I.
0E	D0	iii)	For other addresses simply change YY and XX (Address 05 and 08)

This is not only short but using R0 as address counter so you can enjoy your automatic entering on TEC-1802.
Sincerely yours, S. Takahashi

USING A BAUDOT TELETYPE AS AN OUTPUT DEVICE FOR AN 1802-BASED SYSTEM

by Brian Millier
(Halifax, NS)

The low cost of a CPU card, such as those offered by Tektron or Netronics makes the 1802 chip a natural for anyone who wishes to ease into the microprocessor field without too great a cash outlay. However, before any serious programming can be done some form of terminal must be obtained, as loading via a hex keypad and reading data from LEDs or a 7 Segment Display is rather slow.

At this point the cost of the system will likely soar as a Video Display or a printer will cost upwards of five hundred dollars. Even a reconditioned Model 33 Teletype which is an ASCII machine and will interface with the 1802 hardware and software available, costs over one thousand dollars.

A low cost alternative for those who can get their hands on one, is a Model 15 or 19 Teletype unit. These units have long been removed from commercial service and can often be obtained from ham radio operators for a very modest amount. As they are Baudot machines they require some software modifications if they are to be used on a system that communicates in the ASCII code. This, however is not difficult to implement.

A Baudot machine is however lacking in two areas that cannot easily be overcome:

1) The keyboard is a rather slow mechanical one that is awkward to use; i.e. it must be shifted for letters-figures changes. Also there are no provisions for many of the arithmetic signs used in BASIC, for example.

2) Some useful ASCII characters that might be found in the output string, are not available on the printer. Examples of this are arithmetic signs such as plus, multiplication sign, equals, less than, greater than, etc.

As a result of the above, it was felt that it would be better to use a low cost ASCII keyboard-Encoder chip set configured as a parallel input rather than using the Teletype keyboard as a Serial input device. That looked after 1) above.

The second limitation can be tackled by assigning some of the extra characters such as ! . and dollar sign to the missing arithmetic signs. This usage will soon become quite natural to the user.

The ASCII to Baudot conversion can be performed by a hardware circuit, and indeed many hams have gone this route. However it is more natural to do this in software in a computer system as no additional parts are required and the memory that it takes is minimal.

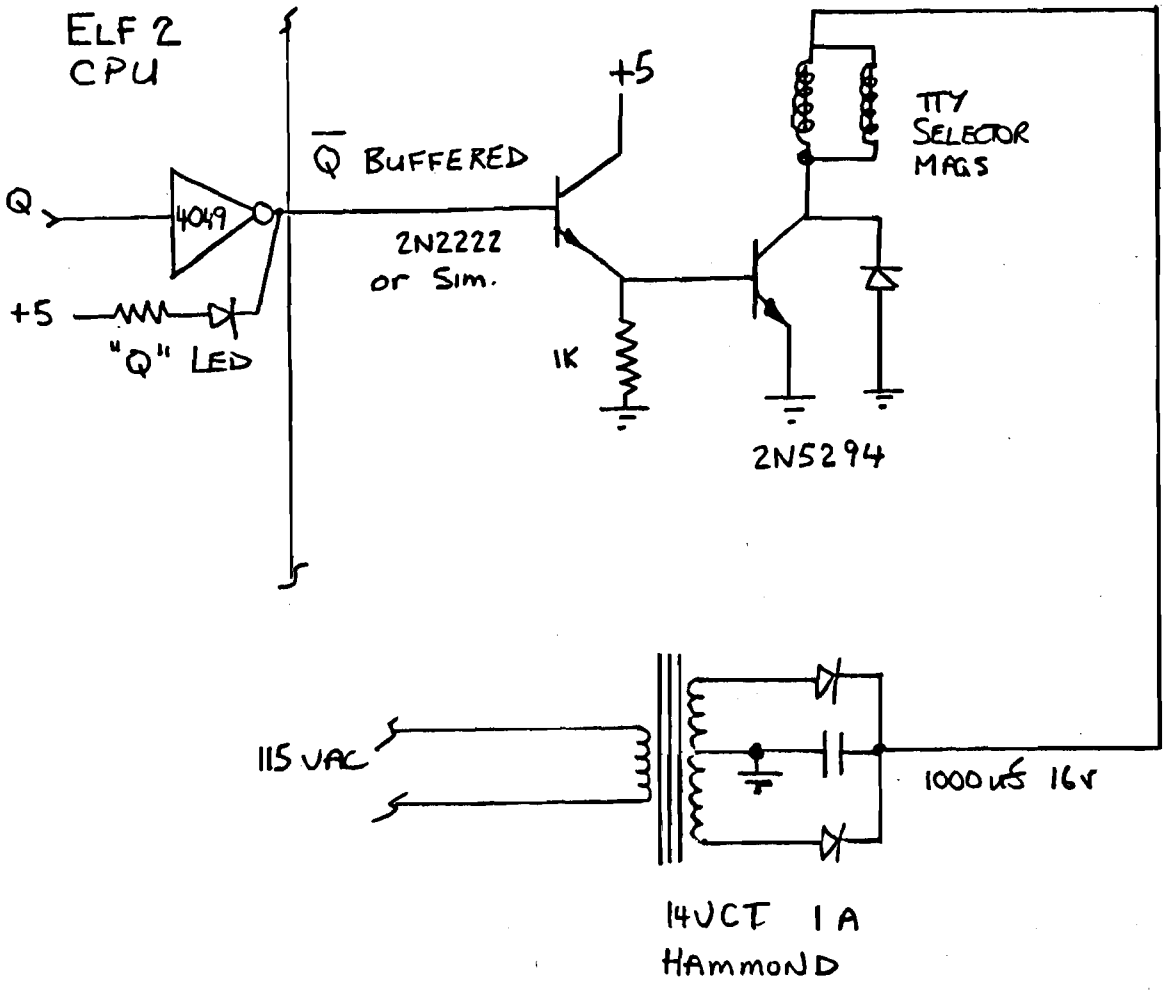


FIG 1

THE HARDWARE

The first thing that must be done is make sure that your Teletype is working well. Details of the operation of the machine are available from many sources. The Sept. 1977 issue of 73 Magazine is largely devoted to TTY, and although there are more comprehensive sources available, I was able to gain insight into the operation of the machine well enough with this issue to get the machine running.

The Selector magnets (on the L.H.S. of machine) should be wired in parallel, rather than in series as this will allow them to work on a low voltage "loop" supply. I am using a simple 10 volt D.C. supply, as shown in figure 1.

Current to run the magnets in this configuration is roughly 60 M.A. This can be handled by any Plastic Tab type transistor with a dissipation of about 5 watts and a breakdown voltage more than 50 or 60 volts. The Reverse-biased diode from coll.-emitter is to suppress the spike produced by the magnets as the field collapses when they are de-energized. I have configured this circuit to be in the "marking" state (machine locked up when no data present) when Q is equal to "0" which is the state that the 1802 will be in after a Reset. Therefore, whatever program is running, it is not necessary to initialize the Q line to lock up the TTY.

I have not encountered any problems in running the magnets directly: i.e. no opto-isolator between the CPU and TTY.

Note that most references to TTY in ham publications will show a high voltage (in the vicinity of 120 volts D.C.) loop supply. This is not required in this case and would only complicate the switching design.

The speed of the Teletype machine that you have must be determined before the software delay loop is written. This can be done by wiring the keyboard contacts into the loop and running the machine in the "local" mode. If an oscilloscope is connected across the keyboard contacts and the space bar is depressed, the single "1" data bit in the middle of the 5 bit data stream will be easy to distinguish and measure. A 22 ms pulse corresponds to a 60 WPM machine, 20 ms equals 66 WPM, 17.57 ms equals 75 WPM.

While the TTY is connected in the local loop check to see whether the machine unshifts (goes from figs-to lets) when the space bar is depressed, or for that matter after a carriage return. I am not familiar with other machines but the model 19 that I have will unshift on space (UOS). This must be taken into account in the software program or the computer will "lose track" of the shift condition of the TTY machine, and print incorrectly.

THE SOFTWARE

The software program to run the Teletype was developed to be compatible with Pittman Tiny BASIC 1802. It is in the form of SCRT type subroutines and can of course be utilized by other programs, such as machine language programs developed by the user.

There are several considerations that arise in developing the necessary software program. The important ones are as follows:

1) The ASCII output of the BASIC program (or users program) must be converted to Baudot. The ASCII output is parallel and is

converted as such to Baudot. This is most easily accomplished by a look-up table. The parallel Baudot data could then be serialized by a UART but it is rather easy to implement this in software so that is the method that I have chosen to use. In the Baudot data I have imbedded a "1" in the L.S.B. in all cases. This is a loop sustaining bit and eliminates the need for a register allocated for use as a counter. (Tiny Basic uses most of the registers, making this a consideration)

2) The software routine must keep track of the nature of the last character printed: i.e. whether it was a letter or a figure. This is done by looking at the M.S.B. of the ASCII char. i.e. the 7th bit. If it is a "0" thechar. is a figure; a "1" indicates a letter. Therefore upon receipt of an ASCII character, the program determines what the present char. is and compares it with the data stored in Register C which contains the nature of the last character sent. If a difference exists a routine which outputs the correct "Figs" or "letters" character takes over, and then returns to the main print program to output the character given it by the main program. After this character is printed, its nature is stored in the Register C for use by the next character that is inputted. The character "space" is trapped before it enters the above routine as it looks like a figure and stores a code in Register C as such, but when it is printed it will unshift the TTY carriage. This has the effect of printing letters in place of numbers if there is a space between them. The "space" character is handled differently in the routine and always results in a letters code being placed in Register C after it is printed.

Certain control codes outputted by Tiny Basic as well as pad characters will also cause the output routine to lose track of the shift condition unless taken care of by the output routine. By setting location 0111 in Tiny to 02 as opposed to the 82 code in the original program, the pads will come out as nulls. The look-up table should code ASCII nulls to a "Figs" character (hex 37). I have set all unused ASCII code locations to the code for Baudot "blank" (hex 01) It appears that Tiny outputs some control characters that are not mentioned in the text, and which will confuse the output routine at times if this coding is not done.

3) This software program uses several registers to store temporary data during the I/O routines. I have chosen to use registers not used by Tiny BASIC 1802. While the text that comes with Tiny states that Register F.0 is not used by the interpreter, I have been unable to have any luck using this register for a purpose such as storing the nature of the last character, which requires that the register is left unaltered between I/O calls. It is therefore used for ASCII character storage through the I/O routine.

If the program is loaded exactly as given it may be tested as follows

Address	Data	
0000	C0 0E 00	Long jump to Initialization (after initialization program will jump to 0004)
0003	XX Don't care	
0004	D4 0E 53	Call Ineee
0007	30 04	Repeat

This will print out the characters typed into the ASCII keyboard.

PARALLEL ASCII KEYBOARD INPUT

I have used a standard ASCII KEYBOARD, G.I. AY5-3600 Encoder and two 4016 quad transmission gates to form the input system.

The Keyboard Encoder Chip's "data ready strobe" is connected to the 1802's $\overline{EF\ 3}$ line and is tested in software. The ASCII data is gated to the bus through the two 4016 quad transmission gates. These are enabled by the N 1 line of the 1802. A "6A" instruction will cause the keyboard entry to be loaded into the Accumulator and M (R X).

System Initialization For Input-Output Subroutines

The following is a system initialization which will be required if the Monitor in your system does not support the SCRT (Standard Call and Return Technique) system of calling sub-routines as set out in the R.C.A. 1802 User's Manual. This is only required if the user wishes to use these routines as part of machine language programs of his own; initialzation is already present in Pittman Tiny Basic and it will run properly without this initialization being entered in memory.

Assuming a 4K memory, maximum space will be left for user's Basic programs if the following scheme is used;

0E00-0E52	Inilialization (if required, per above)
0E53-0EDB	Ineee, Outee, Break subroutines
0F00-0FFF	ASCII-BAUDOT look up Table, Stack

Basically, more than $\frac{3}{4}$ of the last page is not required by the look-up table and could be used for other purpose. I have chosen to leave it filled with the code for a Baudot "blank".

System Initialization

```
0E00  F8 0E       LDI  0E      ; Set this program to run with
  02  B6          PHI  R6      ; R6 as the program counter
  03  B7          PHI  R7
  04  F8 08       LDI  08
  06  A6          PLO  R6
  07  D6          SEP  R6
  08  F8 26       LDI  26      ; Set R7 to point to table of
  0A  A7          PLO  R7      ; initial values.
  0B  47          LDA  R7
  0C  B5          PHI  R5      ; Initialize R5 (SCRT Return
  0D  47          LDA  R7      ; routine pointer)
  0E  A5          PLO  R5
  0F  47          LDA  R7      ; Initialize R4 (SCRT Call
  10  B4          PHI  R4      ; routine pointer)
  11  47          LDA  R7
  12  A4          PLO  R4
  13  47          LDA  R7      ; Initialize R3 (System program
  14  B3          PHI  R3      ; Counter)
  15  47          LDA  R7
  16  A3          PLO  R3
0E17     47       LDA  R7      ; Initialize R2 (System stack
  18     B2       PHI  R2      ; pointer)
  19     47       LDA  R7
  1A     A2       PLO  R2
```

```
1B      47          LDA  R7       ; Initialize R1 (Interrupt Service
1C      B1          PHI  R1       ; pointer)
1D      47          LDA  R7
1E      A1          PLO  R1
1F      47          LDA  R7       ; Initialize R0 (DMA pointer)
20      B0          PHI  R0
21      47          LDA  R7
22      A0          PLO  R0
23      E2          SEX  R2       ; Set X Register to R2
24      7A          REQ           ; Put serial output in
                                  ; Mark state.
                                    Convention Mark equal 0
                                               Space equal 1
                                          -on Q line
25      D3          SEP  R3       ; Go to 0004 At this location at
                                    the beginning of memory user
                                    can select either a long jump
                                    to Basic (0100) or to a Moniter
                                    or other machine language program.
```

Table of Initial Register Values

```
26      0E45        TAB 1         ; SCRT Return routine start address
28      0E33                      ; SCRT Call Routine start address
2A      0004                      ; Jump to Basic or Monitor address
2C      0FFA                      ; Stack location
2E      0000                      ; Interrupt service routine addr.
30      0FFA                      ; DMA storage initial value
32      D3          SEP R3        ; Go to Called subroutine; leave
                                  ; R4 pointing to entry of Call
CALL                              ; subroutine
33      E2          SEX  R2       ; Point to Stack
34      BF          PHI  RF       ; Save D
35      96          GHI  R6       ; Save R6 on Stack
36      73          STXD          ;
37      86          GLO  R6       ;
38      73          STXD
39      93          GHI  R3       ; Copy R3 into R6 to save return
3A      B6          PHI  R6       ; address
0E3B    83          GLO  R3       ;
3C      A6          PLO  R6       ;
3D      46          LDA  R6       ; Load the subroutine address
3E      B3          PHI  R3       ; into R3
3F      46          LDA  R6
40      A3          PLO  R3
41      9F          GHI  RF       ; Restore D
42      30 32       BR   32       ; Branch to 32, Exit point for
                                  ; call subroutine
44      D3          SEP  R3       ; Return to main program; leave
                                  ; R5 pointing to entry of Return
RETURN                              subroutine
0E45    BF          PHI  R7       ; Save D
46      96          GHI  R6       ; Copy R6 into R3 (R6 contains
47      B3          PHI  R3       ; return address)
48      86          GLO  R6
49      A3          PLO  R3
```

```
  4A  E2          SEX  R2       ; Point to stack.
  4B  12          INC  R2       ; Pop stack
  4C  72          LDXA          ; Restore the saved old R6
  4D  A6          PLO  R6       ; into R6
  4E  F0          LDX
  4F  B6          PHI  R6
  50  9F          GHI  RF       ; Restore D
  51  30 44       BR   44       ; Branch to 44, exit point for
                                ; return subroutine
```

Input-Output subroutines for Tiny Basic 1802

```
INEEE
0E53  36 53       B3   53       ; Wait til ASCII Key pressed
  55  3E 55       BN3  55       ; (Keyboard strobe to EF3)
  57  6A          INP  2        ; ASCII data to D, m (R2)
  58  FE FE       SHL,SHL       ; Shift out 2 M.S.B.
  5A  3B 63       BNF  63       ; If DF equal to 0 go to 63
  5C  FE          SHL           ; Shift out M.S.B.
  5D  3B 63       BNF  63       ; If DF equal 0 go to 63
  5F  F8 20       LDI  20       ;
  61  F5          SD   R2       ; Subtract 20 from ASCII char.
                                ; (Change L.C. to U.C.)
  62  38          SKP           ; Skip
  63  F0          LDX  R2       ; ASCII char. on stack to D.
  64  D4 0E 6B    CALL OUTEE
  67  D5          RETURN        ; Return to caller
  68  C4 C4 C4    NOP           ; NOP's for spacing

OUTEE
0E6B  AF          PLO  RF       ; Save ASCII char at RFo
  6C  FB 20       XRI  20       ; Compare char, "space"
  6E  32 96       BZ   96       ; Same? Yes-go to 96
  70  8F          GLO  RF       ; Get ASCII char.
  71  FE FE       SHL,SHL       ; Shift out 2 M.S.B.
  73  3B 8A       BNF  8A       ; If DF equal to 0, go to 8A
  75  8C          GLO  RC       ; Get RC (let-figs register)
  76  FB FF       XRI  FF       ; Compare RC, FF
  78  32 96       BZ   96       ; Same? yes-go to 96
  7A  F8 7A       LDI  7A       ; Load 7A to D
  7C  D4 0E 9F    CALL PRINT
  7F  30 96       BR   96       ; Go to 96
  81  8F          GLO  RF       ; Get ASCII char.
  82  D4 0E 9F    CALL PRINT
  85  F8 00       LDI  00       ; Load 00
  87  AC          PLO  RC       ; to RC0
  88  8F          GLO  RF       ; Get ASCII char.
  89  D5          RETURN        ; Return to caller
  8A  8C          GLO  RC       ; Get RC (let-figs register)
  8B  FB 00       XRI  00       ; Compare RC, 00
  8D  32 81       BZ   81       ; Same? yes-go to 81
  8F  F8 6D       LDI  6D       ; Load 6D to D.
  91  D4 0E 9F    CALL PRINT
0E94  30 81       BR   81       ; Go to 81
  96  8F          GLO  RF       ; Get ASCII char.
  97  D4 0E 9F    CALL PRINT
  9A  F8 FF       LDI  FF       ; Load FF
  9C  AC          PLO  RC       ; to RCo
  9D  30 88       BR   88       ; Go to 88
```

```
PRINT
0E9F  AE          PLO  RE      ; Save ASCII char at REo
  A0  F8 0F       LDI  0F      ; Load 0F
  A2  BE          PHI  RE      ; to RE 1
  A3  C4 C4       NOP  NOP     ; NOP's -extra space
  A5  0E          LDN  RE      ; Load Baudot char. from look-
                               ; up Table-pointed to by RE.
  A6  FE FE       SHL  SHL     ; Shift out 2 M.S.B.
  A8  7B          SEQ          ; Set Q "start bit"
                               ; Convention Mark-"0"
                               ;            Space-"1"
                               ; on Q line
  A9  D4 0E C5    CALL DELAY
  AC  FE          SHL          ; Shift out M.S.B.
  AD  32 BA       BZ   BA      ; If D equals o then go to BA
  AF  3B B4       BNF  B4      ; If DF equals o then go to B4
  B1  7A          REQ          ; Reset Q
  B2  30 B5       BR   B5      ; Go to B5
  B4  7B          SEQ          ; Set Q
  B5  D4 0E C5    CALL DELAY
  B8  30 AC       BR   AC      ; Go to AC
  BA  7A          REQ          ; Reset Q "Stop bit"
  BB  D4 0E C5    CALL DELAY   ; Stop bit two units duration
  BE  D4 0E C5    CALL DELAY   ;
  C1  D5          RETURN       ; Return to caller
  C2  C4 C4 C4    NOP          ; NOP's-extra spaces
DELAY
0EC5  BC          PHI  RC      ; Save Baudot char at RC
  C6  F8 C8       LDI  C8      ; Delay constant for 75 WPM
                               ; Teletype and 1.79 MHZ clock
  C8  FF 01       SMI  01      ; Subtract 1 from D
  CA  C4 C4 C4    NOP          ; NOP's for timing
  CD  C4 C4       NOP          ; NOP's for timing
  CF  3A C8       BNZ  C8      ; D equals 0, No-go to C8
  D1  9C          GHI  RC      ; Restore Baudot char to D
  D2  D5          RETURN       ; Return to caller
  D3  C4          NOP          ; NOP-extra space
BREAKSUB
0ED4  3F D9       BN4  D9      ; If EF4 equals 0 go to D9
                               ; (Break switch connected to
                                  1802 EF 4 line)
  D6  FD 00       SDI  00      ; Set DF equal to 1 (Break cond.
                               ; in Tiny Basic)
  D8  D5          RETURN       ; Return to caller
  D9  FC 00       ADI  00      ; Set DF equal to 0 (No Break)
  DB  D5          RETURN       ; Return to caller
```

ASCII TO BAUDOT LOOK-UP TABLE

Address	Data	(ASCII)	(Baudot)
0F00	37	nul	"figs"
0F07	29	bel	Bell
0F0A	11	LF	line Feed
0F0D	05	CR	Carriage Ret.
0F20	09	SP	Space
0F21	2D	!	!
0F22	23	"	"
0F28	3D	(	(
0F29	13	)	)
0F2A	0F	*	. (no * avail)
0F2B	2D	+	; (no + avail)
0F2C	0D	,	,
0F2D	31	-	-
0F2E	0F	.	.
0F2F	2F	/	/
0F30	1B	0	0
0F31	3B	1	1
0F32	33	2	2
0F33	21	3	3
0F34	15	4	4
0F35	03	5	5
0F36	2B	6	6
0F37	39	7	7
0F38	19	8	8
0F39	07	9	9
0F3A	1D	:	:
0F3B	1F	;	;
0F3C	0F	<	. (no < avail)
0F3D	25	=	(no = avail)
0F3E	0F	>	. (no > avail)
0F3F	27	?	?
0F41	31	A	A
0F42	27	B	B
0F43	1D	C	C
0F44	25	D	D
0F45	21	E	E
0F46	2D	F	F
0F47	17	G	G
0F48	0B	H	H
0F49	19	I	I
0F4A	35	J	J
0F4B	3D	K	K
0F4C	13	L	L
0F4D	0F	M	M
0F4E	0D	N	N
0F4F	07	O	O
0F50	1B	P	P
0F51	3B	Q	Q
0F52	15	R	R
0F53	29	S	S
0F54	03	T	T
0F55	39	U	U
0F56	1F	V	V
0F57	33	W	W
0F58	2F	X	X
0F59	2B	Y	Y
0F5A	23	Z	Z

Control Characters

OF13	37	DC 1	"figs"
OF91	37	DC 3	"figs"

NOTE: Other locations in memory not listed (in page OF that is) should be coded with the code for Baudot "blank" i.e. 01.
This insures that any ASCII characters that Tiny Basic might output that are not available on a Model 15 Teletype keyboard will not result in illegal codes being presented to the Teletype printer.

A MOUSE TRAP GAME FOR PIXIE GRAPHICS

by Dan Rubis
St. Clair Shores. Mich

Tom; ... Enclosed find a mouse trap game program which will work for anyone having Pixie Graphic capability (1861 chip). It uses less than 256 bytes. A speaker should be connected to the Q-line as shown in Sept. 76 issue of "Popular Electronics". The speed of the mouse can be varied by the input byte. The Input switch controls the trap. There is on-screen scoring and sound effects. Sincerely, Dan Rubis

Register Useage for Mouse Trap Game

R0	Program counter
R1	Interrupt
R2	Stack
3	Main
R4	Spare
R5	Spare
R6	# of Lines
R7	# of Bytes
R8	Store 1st Byte on Line
R9	Store Address of Fisrt Line
RA	Store Address of First Line
RB	Freq.
RC	Mouse's Score
RD	Your Score
RE	Time Register for Speed or Mouse
RF	Game Board Pointer & Searcher & End of Game Test

```
                        ;    MOUSE TRAP  - A GAME FOR PIXIE GRAPHICS
                        ;
                        ;       WRITTEN BY DAN RUBIS
                        ;
                        ;    INITIALIZATION SECTION

0000  90                        GHI     R0      ; INITIALIZE - HIGH
0001  B1                        PHI     R1      ;               BYTES
0002  B2                        PHI     R2      ;                OF
0003  B3                        PHI     R3      ;                 SOME
0004  B4                        PHI     R4      ;                  REGISTERS
0005  F8 2D                     LDI     MAIN    ; INITIALIZE MAIN
0007  A3                        PLO     R3      ;  PROGRAM P-COUNTER
0008  F8 BD                     LDI     STACK-1 ; INITIALIZE STACK
000A  A2                        PLO     R2      ;  POINTER
000B  F8 11                     LDI     INTRPT  ; INITIALIZE INTERRUPT
000D  A1                        PLO     R1      ;  HANDLER P-COUNTER
000E  D3                        SEP     R3      ; GO TO MAINLINE

                        ;    INTERRUPT HANDLER

000F  72                INTRTN: LDXA            ; RESTORE D REG.
0010  70                        RET             ; RETURN FROM INTERRUPT
0011  22                INTRPT: DEC     R2      ; R2 IS THE STACK POINTER
0012  78                        SAV             ; SAVE T REG ON STACK
0013  22                        DEC     R2
0014  52                        STR     R2      ; SAVE D REG. ON STACK
0015  C4                        NOP             ; SYNC
0016  C4                        NOP             ;  SYNC
0017  C4                        NOP             ;   SYNC
0018  F8 00                     LDI     #00
001A  B0                        PHI     R0
001B  F8 00                     LDI     #00
001D  A0                        PLO     R0
001E  80                REFRSH: GLO     R0      ; GET LINE START ADDRESS
001F  E2                        SEX     R2
0020  E2                        SEX     R2      ; NOP
0021  20                        DEC     R0
0022  A0                        PLO     R0
0023  E2                        SEX     R2      ; NOP
0024  20                        DEC     R0
0025  A0                        PLO     R0
0026  E2                        SEX     R2      ; NOP
0027  20                        DEC     R0
0028  A0                        PLO     R0
0029  3C 1E                     BN1     REFRSH  ; LOOP TIL "DISPLAY STATUS" LOW
002B  30 0F                     BR      INTRTN

                        ;    MAIN LINE

002D  E2                MAIN:  SEX     R2      ; TURN ON TV
002E  69                        INP     1
002F  F8 C0 AF                  +DLDI   BOARD,R15       ; RF POINTS TO DISPLAY AREA
0032  F8 00 BF
0035  F8 00             BLANK:  LDI     #00     ; BLANK OUT BYTE IN
0037  5F                        STR     RF      ;  DISPLAY AREA
0038  8F                        GLO     RF      ; CHECK FOR END OF BOARD
0039  FB FF                     XRI     #FF     ;  ( IE. END OF PAGE )
003B  32 40                     BZ      MOUSE   ;    - YES - CONTINUE ON
003D  1F                        INC     RF      ;    - NO  - INC. TO NEXT
003E  30 35                     BR      BLANK   ;      BYTE AN BACK FOR NEXT
0040  F8 D0             MOUSE:  LDI     #D0     ; SET REG. RF TO POINT
0042  AF                        PLO     RF      ;  TO MOUSE PICTURE
0043  F8 01                     LDI     #01     ;   AND STORE THE
0045  5F                        STR     RF      ;    MOUSE
0046  F8 CB                     LDI     #CB     ; SET REG. RF TO POINT
0048  AF                        PLO     RF      ;  TO GAME BOARD PICTURE
0049  F8 05                     LDI     #05     ;   AND STORE PART OF
004B  5F                        STR     RF      ;    GAME BOARD
004C  F8 DB                     LDI     #DB     ; AND THEN
004E  AF                        PLO     RF      ;  DO SECOND
004F  F8 05                     LDI     #05     ;   PART OF
0051  5F                        STR     RF      ;    BOARD
0052  F8 00                     LDI     #00
0054  B9                        PHI     R9
0055  BA                        PHI     RA
0056  BC                        PHI     RC
0057  BD                        PHI     RD
0058  F8 DD                     LDI     #DD     ; INITIALIZE SCORE
005A  AD                        PLO     RD      ;  BOARD
```

```
0058  F8 DE             LDI     #DE
005D  AC                PLO     RC
005E  F8 D0     LOOP:   LDI     #D0     ; INITIALIZE ADDRESS
0060  A9                PLO     R9      ;  OF TRACK
0061  AA                PLO     RA
0062  F8 07             LDI     #07     ; R7 = NUMBER OF BYTES - 1
0064  A7                PLO     R7      ;  IN TRACK
0065  09                LDN     R9      ; STORE FIRST BYTE ONLINE
0066  B8                PHI     R8
0067  76                SHRC            ;  ROTATE RIGHT
0068  19                INC     R9      ; MOVE POINTER TO NEXT BYTE ONLINE
0069  09                LDN     R9
006A  76                SHRC            ;  ROTATE RIGHT
006B  59                STR     R9      ; RESTORE BYTE
006C  27                DEC     R7      ; ONE LESS BYTE
006D  6C                INP     4       ; READ MOUSE SPEED FROM KEYBOARD
006E  AE                PLO     RE      ;  AND SAVE IN TIME REGISTER
006F  2E        DELAY:  DEC     RE      ; DELAY LOOP
0070  8E                GLO     RE
0071  3A 6F             BNZ     DELAY
0073  87                GLO     R7      ; ALL BYTES SHIFTED ?
0074  3A 68             BNZ     #68
0076  98                GHI     R8      ; SHIFT FIRST BYTE ONLINE
0077  76                SHRC
0078  5A                STR     RA      ;  AND STORE
0079  37 7D             B4      SKIP    ; CONTINUE SHIFTING
007B  30 5E             BR      LOOP    ;  UNTIL 'IN' ON
007D  F8 D3     SKIP:   LDI     #D3
007F  AF                PLO     RF
0080  0F                LDN     RF      ; TEST IF MOUSE
0081  FB 02             XRI     #02     ;  IS CAUGHT
0083  3A 91             BNZ     #91
0085  F8 01             LDI     #01     ; SCORE 1 FOR YOU
0087  5D                STR     RD
0088  8D                GLO     RD
0089  FC 08             ADI     #08
008B  AD                PLO     RD
008C  F8 20             LDI     #20     ; SET BUZZER
008E  AB                PLO     RB      ;  FREQUENCY
008F  30 9B             BR      BUZZER  ;  AND BUZZZZ
0091  F8 01             LDI     #01     ; SCORE 1 FOR THE MOUSE
0093  5C                STR     RC
0094  8C                GLO     RC
0095  FC 08             ADI     #08
0097  AC                PLO     RC
0098  F8 FF             LDI     #FF     ; SET BUZZER
009A  AB                PLO     RB      ;  FREQUENCY
009B  7B        BUZZER: SEQ
009C  8B                GLO     RB
009D  FF 01     BUZON:  SMI     #01
009F  3A 9D             BNZ     BUZON
00A1  7A                REQ
00A2  8B                GLO     RB
00A3  FF 01     BUZOFF: SMI     #01
00A5  3A A3             BNZ     BUZOFF
00A7  37 9B             B4      BUZZER  ; BUZZ WHILE 'IN' IS ON
00A9  F8 FD             LDI     #FD     ; END OF GAME ?
00AB  AF                PLO     RF
00AC  4F                LDA     RF
00AD  FB 01             XRI     #01
00AF  32 B8             BZ      HALT
00B1  0F                LDN     RF
00B2  FB 01             XRI     #01
00B4  32 B8             BZ      HALT
00B6  30 5E             BR      LOOP    ; LOOP BACK FOR ANOTHER PASS
00B8  30 B8     HALT:   BR      HALT    ; END OF GAME HANG UP
00BA  00 00             .DBYTE  0000,0000
00BC  00 00

                ;    STACK STARTS HERE
00BE            STACK:

                ;    GAME BOARD STARTS HERE
      = 00 C0           .ORG    #C0
00C0            BOARD:
                        .END
```

USER SYMBOL TABLE

```
BLANK 0035   BOARD 00C0   BUZOFF 00A3   BUZON 009D   BUZZER 009B   DELAY 006F   HALT  00B8   INTRPT 0011
INTRTN 000F  LOOP  005E   MAIN   002D   MOUSE  0040   REFRSH 001E   SKIP  007D   STACK 00BE
```

PROGRAM SIZE = 00C0
0 ERRORS DETECTED

1802 PROGRAMMER'S NOTEBOOK By David R. Wright, Ames, Iowa

Dear Sir:

Your publication was mentioned in Ross Wirth's 1802 Software Exchange as someone to send information to on the RCA 1802 Microcomputer. I have enclosed a complementary copy of some material you might find useful. I plan to have additional material in the future. Additional copies are available at 25¢ per page SASE or $1.00 PPD for the packet. Sincerely, David R. Wright, 128 Campus Ave., Ames, Iowa 50010, April 23, 1978

Hardware requirements: These programs were developed on the COSMAC ELF II. The computer has a hexadecimal keyboard, 256 bytes of memory and utilizes I/O Port 4. Programs are initiated at memory location ØØ with the program counter register (P=Ø) and the remaining 15 registers on the 1802 are not dedicated. The computer uses a 3.579545 MHz Crystal, whose frequency is divided in half before entering the 1802 Clock input. Other computer systems may use hardware in a different configuration. This may mean some software modification. If this is the case, Assembly listings of these programs are available.

(1) Random Hex-Number. Displays a random hexadecimal number (ØØ to FF) each time the input key is pressed. Because the computer can count faster than the uncertainty in the user's reaction time and each count is exactly the same number of machine cycles from the last count, the output is a random number.

F8 ØD A1 12 3F Ø3 82 51 E1 64 21 3Ø Ø3 ØØ

(2) Seconds Clock. Displays decimal seconds (ØØ to 59) for the 1802 clock frequency of 1.789773 MHz. For frequencies other than this, calculate th number of machine cycles per second (M). M is one-eighth the 1802 clock frequency (M=223,721.6 for the ELF). Next calculate X for the equation: M=6(X-256)+10(255)+44(9/10)+60(1/12)+64(1/60). Round X to the nearest whole number and convert it to hexideciamal. Of that number, place the most significant byte in location 28 (contents 9Ø) and the least significant byte in location 25 (contents F6). If you don't know the frequency, use sucessive approximation. The larger the value of X you choose, the slower the clock will run.

F8 FF A2 B2 F8 33 A3 12 82 FB ØA 3A 1C F8 ØØ A2 92 FC 1Ø B2 92 FB 60 3A 1C F8 ØØ B2 82 53 E3 92 F4 53 64 23 F8 F6 A1 F8 9Ø B1 21 91 3A 2A 81 3A 2A 3Ø Ø7 ØØ

(3) Hex Addition. Program performs single precision (1 byte) addition. Starting or restarting the program clears the accumulator. Pressing the input switch loads the first byte. Pressing the input switch again turns on the Q-Lamp and reveals the contents of the accumulator. Repeating this cycle will sum the next input to the contents of the accumulator.

F8 1F A2 F8 2Ø A1 F8 ØØ 51 3F Ø9 7A E2 6C 64 22 37 1Ø 3F 12 E1 F4 73 11 64 21 7B 37 1B 3Ø Ø9 ØØ ØØ

(4) Hex Multiplication. Program performs single precision multiplication. Press the input switch to load the first number and then the second. Press the input switch again and the Q-Lamp will come on with the answer. The cycle is then repeated.

F8 43 A4 F8 44 A5 F8 45 A6 3F Ø9 7A E4 6C 64 24 37 1Ø 3F 12 E5 6C 64 25 3, 18 A2 44 24 A1 F8 ØØ A3 F8 Ø8 A7 81 F6 A1 CF 3Ø 3Ø 83 56 82 E6 F4 A3 82 FE A2 27 87 3A 24 3F 37 7B 83 56 E6 64 26 37 3F 3Ø Ø9 ØØ ØØ ØØ

Note: Programs (3) and (4) do not have overflow detection.

Relocation Technique: Any software written in 1802 machine language may be used on any other 1802 system, provided the source program is properly relocated.

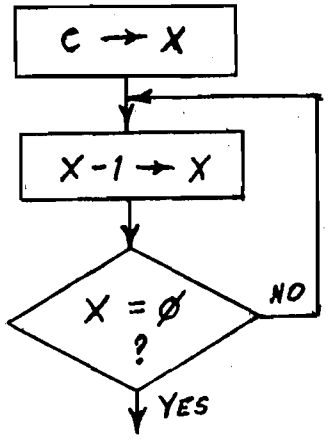
(a) Memory Relocation: The operand of each branch instruction must be changed to reflect the displacement the program has undergone. Initial register values, such as stack and subroutine addresses, must also be changed.

(b) Register Assignment: Some systems have several of the scratch pad registers dedicated to the operating system located in ROM and are therefore unavailable for use by the source program. Also, the program counter (P=?) and stack pointer (X=?) may also have to be reassigned. Make sure all register operations are reassigned when necessary.

(c) Input/Output Ports: If your system uses different I/O Ports than the source program, these instructions must be changed.

One advantage of writing programs in assembly language is that registers, addresses and I/O ports may be referred to symbolically giving the same assembly listing but a different machine code.

(d) Timing Constants: Clocks, frequency counters and music programs use the 1802 system clock for timing. There are eight clock cycles per machine cycle. Thus a machine that operates on a 2-MHz clock runs at 250,000 machine cycles per second. All instructions are two machine cycles in length with the exceptions of the skip, long-branch and continue instructions which are three machine cycles. Programs that require time delays longer than six machine cycles may use timing loops such as the one shown on the right. Any program using a simple timing loop with a timing constant (C) may be calculated using the formula:



$$C = (A/M) + B$$

Where (A) is the number of machine cycles in one transit of the timing loop, (M) is the number of machine cycles per second and (B) is arbitrary. Given A, M and C of the source program, you can calculate B. Given this A, B and M (your machine), you can calculate C for the program to work on your machine.

(5) Minimum Debug and Operating System: User programs begin after location 27.

(a) Operating System:

1) To examine a memory location or sequence, set Ø1 and press the run key, set the memory location and press the input key and the selected memory byte will be displayed. Press the input key again for the next byte of the sequence.

2) To modify memory, set Ø2 and press the run key, set the memory location and press the input key, and load the byte sequence.

3) To run a program, set ØØ and press the run key, set the memory location and press the input key and the program will begin running with R3 as the program counter.

(b) Debug System: RF is a dedicated register.

1) Examine and save the two bytes appearing immediately after the selected breakpoint.

2) Place (3Ø 21) after the selected breakpoint in the program undergoing debugging.

3) Place in location 22 one of the following:

a) C4 to indicate register D.

b) 7E to indicate the data flag DF (Odd, DF=1; Even, DF=Ø).

c) 8N to indicate register RN.Ø.

d) 9N to indicate register RN.1.

N is the register number (N≠ 3 or F).

4) Run the program under test and examine what occurs at your selected breakpoint.

5) Restore the two saved bytes that appeared after the breakpoint.

6) Select a new breakpoint and repeat the process.

```
F8 2Ø AF EF 6C 64 2F 3F Ø7 37 Ø9 32 1D F6 33 11
7B 6C AF 3F 13 37 15 39 1A 6C 64 3Ø 13 6C A3 D3
ØØ EF C4 5F 64 2F 7B ØØ
```

References:

1) 'ETOPS', "Build the COSMAC'ELF' Microcomputer, Part 3", Popular Electronics, March 1977, P. 64.

2) "Jack and the Machine Debug", BYTE, December 1977, P. 91.

Calibrating the 1802 System Clock: If you require applications where accurate timing loops are a must and you don't have a Frequency Counter, your system clock can be calibrated using a software clock and an accurate timing source such as WWV or a crystal-controlled clock or watch. Given the timing constant and equation for your software clock, solve for M (the number of machine cycles per second). Measure T (the time the software clock has run) and ΔT (the accuracy of the software clock). Multiply M by $(T - \Delta T)/T$ to get the value of M for your 1802 system clock. Testing the software seconds program over a period of 24 hours will give an accuracy for your M of 2 machine cycles per second.

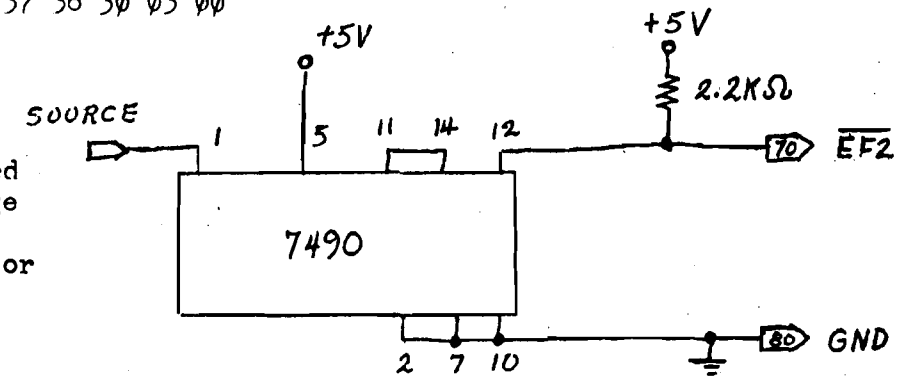
(6) Clock: This clock features reset, hours, minutes and seconds. To reset the clock, set the hour, press the run key, set the minutes, press the input key, set the seconds. To start the clock after reseting it, press the input key. Input ØØ to show the hour, input Ø1 to show the minutes and input Ø2 to show the seconds. For 1802 systems that use frequencies other than M = 223,721.6 machine cycles per second, solve X = (1/14) M + 251.03. Convert X to the nearest whole hexadecimal integer. Place the most significant byte of X in location 48 and place the least significant byte of X in location 4B. The clock should be accurate to about one part in 10,000. (134 bytes)

```
F8 67 A5 F8 66 A3 F8 68 A2 F8 Ø3 A6 E5 6C 64 FA
FØ 52 12 25 FØ FA ØF 52 12 3F 19 37 1B 26 86 3A
ØD F8 6D A2 F8 6E A4 E4 Ø2 F4 52 6Ø F5 3A 47 6Ø
72 52 22 82 FF 67 32 47 FF Ø2 3A 28 22 42 FD 1Ø
32 28 F8 8Ø A4 3Ø 28 F8 3F B1 F8 67 A1 Ø3 FE FC
68 A4 72 F4 55 21 37 62 C4 C4 91 3A 55 E5 64 25
3Ø 21 E3 6C 3Ø 5A Ø2 ØØ ØØ ØØ ØØ ØØ ØØ ØØ Ø1 ØA
ØØ 1Ø 6Ø ØØ Ø1 ØA ØØ 1Ø 6Ø ØØ Ø1 Ø3 Ø1 1Ø 2Ø ØØ
Ø1 ØA ØØ 1Ø 2Ø ØØ
```

(7) Hex Frequency Counter: The first byte of the frequency will appear after the Q-Lamp comes on, pressing the input key will show the second byte of the frequency followed immediately by turning the Q-Lamp off, a one second cycle for the next frequency count and the cycle has been repeated. For this application, the input frequency source uses port 2. A 2.2K ohm pull-up resistor could be used for TTL compatibility. For an 1802 clock frequency of 1.789773 MHz, the counter will handle a minimum pulse width of 626 microseconds at a maximum frequency of 7990 Hz. For other 1802 clock frequencies, use M=10(X-256)+14(255)+10. Convert X to the nearest whole hexadecimal integer and place the most significant byte of X in location ØB and the least significant byte of X in location Ø8. (59 bytes)

```
F8 3A A3 F8 ØØ A1 B1 F8 FE A2 F8 56 B2 3D ØD 7A
35 1Ø 22 92 3A 19 82 32 29 3D 27 11 22 92 3A 23
82 32 29 3D 27 3Ø 1C 3Ø 12 7B 91 53 E3 64 23 3F
2F 81 53 E3 64 23 37 36 3Ø Ø3 ØØ
```

Prescaler in Bi-Quinary Mode. Source is connected to pin 12 of a prior stage 7490 prescaler. Use an independent power supply or low power TTL.



(8) Multiple precision Hexadecimal Adder: This program adds two 8-digit hexadecimal numbers. The accumulator is cleared when the run key is depressed. The 4-byte (8 digit) number is inputed followed by addition to the value in the accumulator, the Q-Lamp comes on and pressing the input key four times gives the sum. (85 bytes)

```
F8 54 A3 F8 04 A1 F8 00 E3 73 21 81 3A 06 F8 04
A1 F8 4D A2 3F 14 E2 6C 64 37 19 21 81 3A 14 F8
04 A1 F8 50 A2 F8 54 A3 03 E2 F4 53 21 22 23 03
E2 74 53 21 22 23 81 3A 2F 7B F8 04 A1 F8 51 A3
3F 40 E3 64 37 44 21 81 3A 40 7A 30 0E 00 00 00
00 00 00 00 00
```

(9) ASCII Decimal to Hexadecimal Converter: An eight-place ASCII decimal number is inputed by pressing the input key eight times. The hexadecimal result requires four depressions of the input key to show the result. (144 bytes)

```
F8 74 A7 F8 8A A1 F8 05 A2 22 82 32 13 F8 00 51
21 30 09 F8 8F A3 F8 08 A2 D7 F8 8A A1 F8 8E A4
F8 04 A5 01 54 21 24 25 85 3A 23 D7 D7 F8 8A A1
F8 8E A4 F8 03 A5 04 E1 F4 73 24 25 04 74 73 85
3A 3A 3F 42 E3 6C 64 23 FA 0F 37 4A 53 F8 8A A1
03 E1 F4 73 3B 5B F0 7C 00 30 53 22 82 3A 19 F8
87 A1 F8 04 A2 7B 3F 66 E1 64 37 6A 22 82 3A 66
7A 30 03 D0 F8 8A A1 F8 03 A6 E1 F0 FE 73 F0 7E
73 26 86 3A 7E 30 73 00 00 00 00 00 00 00 00 00
```

(10) Hexadecimal Calculator: Twelve-digit add, subtract, multiply and divide can be performed. Pressing the run key clears the accumulator. To enter a number requires six presses of the input key followed by the operation code 00 for add, 01 for subtract, 02 for multiply and any other number for divide, a press of the input key and six more presses of the input key for the result. Division is performed with the remainder truncated. There is no overflow indicator. (256 bytes)

```
F8 F2 A3 F8 90 A4 F8 9F A5 F8 B0 A6 F8 C2 A7 F8
D7 A8 E2 D4 F8 7B F8 F9 A2 F8 07 A1 3F 1C 37 1E
6C AD 64 21 81 3A 1C 8D 3A 3F D7 F8 FE 7A F8 06
A1 F8 F3 A2 3F 34 37 36 64 21 81 3A 34 30 15 2D
8D 3A 46 D8 30 2D D4 F2 2D 8D 3A 70 F8 2F A1 FE
D5 3B 56 D7 F2 F8 81 3A 6B F8 06 A1 F8 F8 A2 F8
F2 AE 0E 73 2E 21 81 3A 62 30 2D 21 D6 F8 30 4F
F8 00 A1 D6 FE 3B 86 D5 D8 33 89 D7 F8 FE 81 32
59 D6 F2 21 30 77 11 30 73 03 FC 01 53 30 7E D0
F8 06 A9 40 A2 F8 00 73 29 89 3A 95 30 8F D0 F8
06 A9 F8 F9 A2 F0 76 52 60 29 89 3A A5 30 9E D0
F8 05 A9 40 A2 F0 FE 73 F0 7E 73 29 89 3A B8 30
AF D0 F8 05 A9 40 A2 40 AC 0C F4 73 2C 0C 74 73
29 89 3A CC 30 C1 D0 F8 05 A9 F8 F8 A2 F8 FE AB
0B F5 73 2B 0B 75 73 29 89 3A E3 30 D6 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

ERRATTA

A "ROM Out of RAM", Issue #4, pg. 25: The Editor's Note at the end of the article is incorrect. The memory chips will not drive the data buss if the CPU writes to the memory block when it is write-protected.

B "Cassettes and Computers", Issue #3, pg. 23-28. In Figure 2 on pg. 26, the value of R13 is unspecified. It is 22 Kohms. Also, despite what the accompanying article says, the KC Interface, as shown, has "0"-True Data Input and Output. That is:

Data In	Tone	Data Out
0	2400 Hz	0
1	1200 Hz	1

This arrangement is operable, until you want to exchange recorded tapes with someone else who uses "1"-True Data in his KC Interface. Then you will find the tones on the tape are wrong.

To obtain "1"-True data in and out, do 2 things: (1) Take DATA OUT from pin 2 instead of pin 1 of IC1A (4013). (2) Add an inverter (4049 typical) in the DATA IN line, before IC2A, to invert the data input.

C "KC Interface Test Routine", Issue #3, pg. 29-30. If you do not change the data input and output as shown above, then make the following changes to the Test Software:

LOCATION	OLD INST.	NEW INST.
1A	REQ (7A)	SEQ (7B)
28	SEQ (7B)	REQ (7A)

If you do make the hardware changes shown above, then change the following locations in the software:

LOCATION	OLD INST.	NEW INST.
3A	B2 (35)	BN2 (3D)
41	B2 (35)	BN2 (3D)
46	BN2 (3D)	B2 (35)
53	B2 (35)	BN2 (3D)

D "Hardware Paper Tape Loader", Issue #5, pg. 4-10. Figure #1 was accidently omitted from this article, so here it is now, with our apologies:

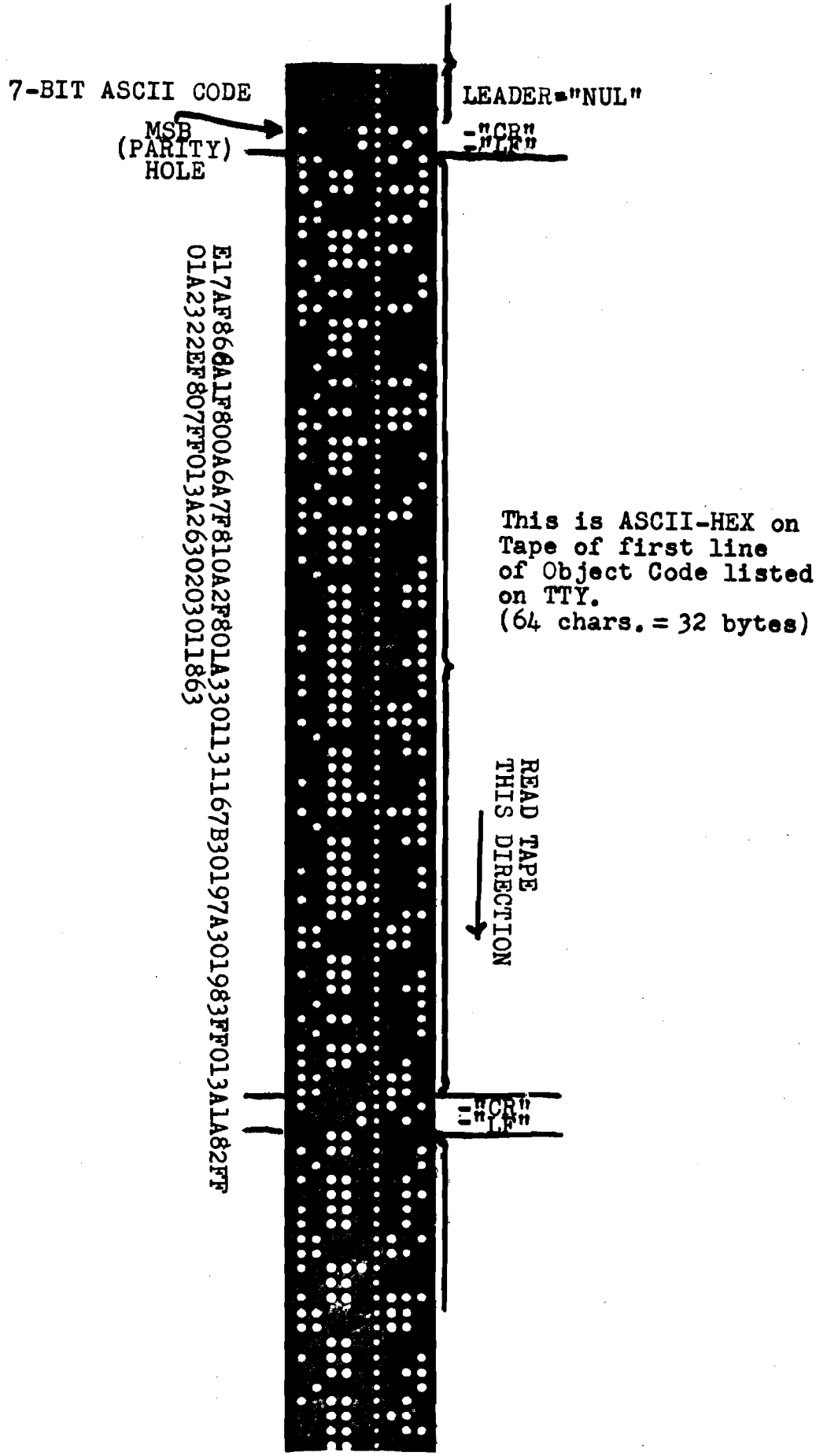


Figure 1: TAPE FORMAT

AN 1802 RAM SYSTEM

by Bernie Murphy
Oakville, Ont.

If your COSMAC 1802 is suffering from lack of memory, this project may be the cure.

Prime design goals for a memory system are:

1. lowest power consumption possible
2. static memory (easy to use)
3. minimum parts
4. low cost

I believe that by using 2114 static RAM chips (4K bits), the above have been achieved.

At the time this article was written, the cost for a prime 2114 was $6.75 ($U.S.). This price is very favourable compared to 4 2102 chips costing $5.00($U.S.).

The circuit in figure 1 has been running successfully for 2 months.

I am currently running a 1 MHZ clock and using 2K out of a possible 4K; expansion above 4K requires another decoder. The circuit should run at 2MHZ without any problems. Speeds above 2 MHZ will probably require buffers on the address lines as well as a faster chip than the 4042 which latches the high portion of the address on the address bus.

By adding a few more gates, write protect and bank disable can easily be added. Once these features are operating, I will document them in another article.

The bus configuration in figure 1 is for the TEC-1802 bus.with the CPU clock brought out on pin 25.

In order that write protect works properly-when using the M key function, the TEK 1802 CPU board requires that the $\overline{\text{MWR}}$ line that goes to pin 29 of the edge connector come from V14 pin 4 and not the $\overline{\text{MWR}}$ pin 35 of the MPU. By cutting the trace from pin 35 of the MPU to pin 29 of edge connector and placing a jumper from U7 pin 20 to pin 29 of the edge connector, this can be accomplished.

Please note that the on-board 256 bytes of RAM has to be removed if the full 4K address space is desired. By cutting V5 on the T-C-1802 CPU board and connecting the $\overline{\text{CE1}}$ pin on this board to one of the $\overline{\text{BANK CS}}$ lines, a 1.25K 2.25K or 3.25K configuration is possible.

If anyone has any questions or comments about my design please contact me: Bernie Murphy, 102 McCrany St., Oakville, Ont., L6H 1H6.

References:

1. User manual for the CD1802 Cosmac Microprocessor, MPM201A, RCA Corporation
2. RCA COS/MOS, SSD-203C, RCA Corporation
3. Lancaster "INS AND OUTS OF VOLATILE MEMORIES", BYTE, November 1975, Pg. 12
4. RCA COSMAC VIP Instruction Manual, RCA Corporation.

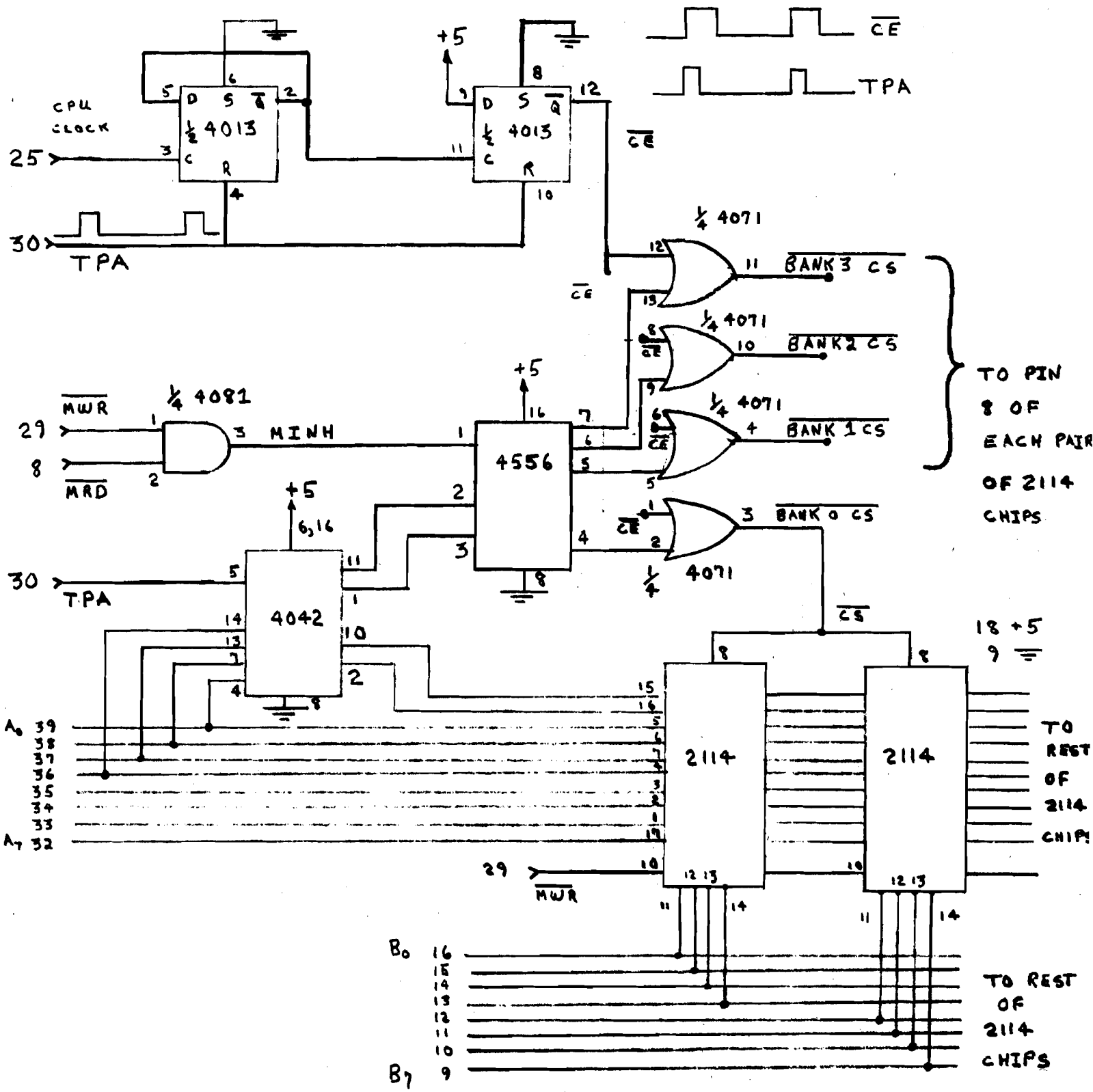


Figure 1. An 1802 RAM System

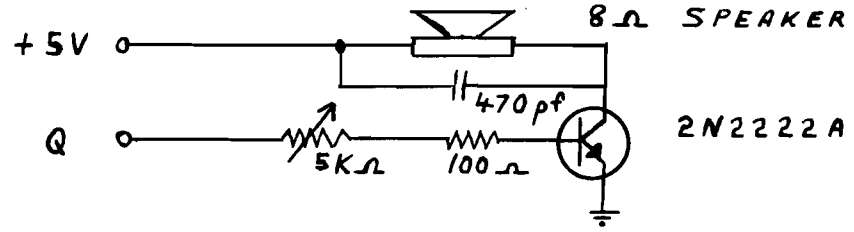
COMBINATION LOCK AND DOOR CHIME

Anthony Tekatch
Caister Centre, Ont.

This program combines the music generation program (IPSO FACTO #1) with some simple sequential logic to produce a combination Door Lock nd Door Chime.

Connections

1. Disconnect your front door bell button from the existing chimes and power transformer, and connect it between ground and EF4. On the TEC-1802, this is in parallel with the "I" pushbutton, so testing can be done with "I" if desired.
2. Connect a simple amplifier and speaker to the "Q" line from the 1802, and mount near your existing chimes. A test circuit can be easily and cheaply constructed as shown:



3. Bolt an electric lock mechanism to your front door, and wire it via an inexpensive relay and transistor driver (as above for speaker) to the MSB of the byte accessed by an OUT 4 instruction. On the TEC-1802, this is the MSB of the DATA leds. The polarity is such that a "1" (led ON) will operate the bolt mechanism. Electric door ocks are available from such manufacturers as NEDCO, and from Edmund Scientific Co.

Operation

The program is continually watching for operation of the door-bell button. Once it sees the first push of the button it begins a series of timing loops, each approximately 1 second long. At the end of each time period, the program checks to see whether the button is pressed or not, depending upon the particular combination which has been programmed in as the right one for opening the door lock mechanism. In the present program, there are 4 time loops, therefore 4 checks of the button are made after the initial push. The present combination is:

First Push	1 secs	2 secs	3 secs	4 secs
1	1	0	0	1

and can be changed easily by changing the instructions at 11, 19, 21, and 29 hex. (The instruction at 09 hex should remain a 3F as long as the door bell button is a N.O. type.)

If at any point in the above combination the program sees the wrong sense of the pushbutton, ie. pushed when it shouldn't be, then it will immediately play the tune programmed into locations 5F to 6F, then go back to the beginning of the program.

If the combination is correct all the way through, the door lock mechanism will be operated for approximately 2 seconds, then the micro tarts again at the beginning.

Note that this program assumes 256 bytes of memory only.

ADDRESS	INSTRUCTION			MNEMONIC		COMMENTS
00	F8	00	START:	LDI	#00	
02	A1			PLO	R1	
03	E1			SEX	R1	
04	64			OUT4		Ensure lock is closed
05	F8	70		LDI	#70	X points at time
07	A3			PLO	R3	delay constant
08	E3			SEX	R3	
09	3F	09		BN4	@	Watch for button pushed.
0B	F0			LDX		YES-get delay constant
0C	B2			PHI	R2	and
0D	22		DELAY1:	DEC	R2	delay for a second
0E	92			GHI	R2	
0F	3A	0D		BNZ	DELAY1	
11	3F	3D		BN4	WRONG	Look for a "1"
13	F0			LDX		YES -
14	B2			PHI	R2	
15	22		DELAY2:	DEC	R2	delay a second
16	92			GHI	R2	
17	3A	15		BNZ	DELAY2	
19	37	3D		B4	WRONG	Look for a "0"
1B	F0			LDX		Yes -
1C	B2			PHI	R2	
1D	22		DELAY3:	DEC	R2	delay a second
1E	92			GHI	R2	
1F	3A	1D		BNZ	DELAY3	
21	37	3D		B4	WRONG	Look for a "0"
23	F0			LDX		Yes -
24	B2			PHI	R2	
25	22		DELAY4:	DEC	R2	delay a second
26	92			GHI		
27	3A	25		BNZ	DELAY4	
29	3F	3D		BN4	WRONG	Look for a "1"
2B	F8	70		LDI	#70	Yes -
2D	A1			PLØ	R1	
2E	E1			SEX	R1	
2F	64			OUT4		Open the lock,
30	C4			NOP		
31	F8	FF		LDI	#FF	
33	B1			PHI	R1	
34	21		DELAY5:	DEC	R1	
35	C4			NOP		Wait a couple of
36	C4			NOP		seconds,
37	C4			NOP		
38	91			GHI	R1	then
39	3A	34		BNZ	DELAY5	
3B	30	00		BR	START	go close the lock.
3D	F8	08	WRONG:	LDI	#08	Enter here if combina-
3F	A1			PLO	R1	tion entered is wrong.
40	F8	5F		LDI	SONG	Point to start of chime
42	A2			PLO	R2	tune (R2), # of
43	F8	20	PLAY1:	LDI	#20	notes (R1), and
45	B3			PLO	R3	length of note (R3)
46	7A		PLAY2:	REQ		

```
47    02        PLAY3: LDN    R2      Get song note
48    A4               PLO    R4
49    24       PLAY4: DEC    R4
4A    23               DEC    R3      Set and reset Q
4B    93               GHI    R3      at a frequency
4C    32 56            BZ     PLAY5   determined by value
4E    84               GLO    R4      of tune note.
4F    3A 49            BNZ    PLAY4
51    31 46            BQ     PLAY2
53    7B               SEQ
54    30 47            BR     PLAY3
56    12       PLAY5: INC    R2      Point to next song note
57    21               DEC    R1
58    81               GLO    R1
59    37 00            B4     START   Button pushed? Yes
5B    32 00            BZ     START   Tune finished? Yes
5D    30 43            BR     PLAY1   No. Continue playing tune
5F             SONG   .BLOCK #11      17 song notes go here
70    80               .BYTE  #80      = 1 second delay constant
```

A DIFFERENT CASSETTE I/O ROUTINE

by Robert G. Edwards
Oak Ridge, TN

Dear Tom,

Several months ago I mentioned to you that I was working on 1802/ cassette interfaces, and that I was not happy with McCormick's 2MHz version from Dr. Dobb's Journal. I have investigated the problem to some extent, and I was aided considerably by your comments. I would appreciate your consideration of the attached 1.79 MHz cassette routines. The memory to cassette routine starts at zero; the cassette to memory starts at fifty (hex). The routines are set up for a one page memory system, so that they read or write the 112 remaining bytes of the 256 byte memory. I used McCormick's one transistor interface between the computer and the recorder, except that the input to the recorder was attenuated so I could use the MIC input. Also, I drove the recorder with "not Q" rather than Q as he did. I have used this routine with great success with two cheap recorders. One is a 1961 Zenith and the other is a hand-held Panasonic dictation recorder. Each will read the other's tapes.

The theory of the cassette to memory routine deserves a little explanation. The strategy is to assume that the two stop bits and the start bit produce a consistent wave-form, independent of what preceeds or follows them. The technique is to sync on some characteristic of this wave form, then set up a string of eight sampling windows. The program counts the number of transsitions in each window (a simulated frequency to voltage converter). This gives a statistical basis for each bit of the data.

There are two parameters in the cassette to memory routine. (They are marked "See Note 2" in the listing at locations 53 and 62 hex.) The constant at "CASREAD" determined the criterion for establishing the synchronization for the byte to be read. In this program, we are looking for an up part of a tone cycle 64 machine cycles long.

The constant at "SDELAY" is used to position the string of eight sample windows. To determine the proper values for these two constants, we need to modify the program so that it records the transitions counter (R5) for a dozen or so bits after a sync. To accomplish this, change the constant at "NOBITS" to 24 (hex), change the three instructions starting at "SETD" to BR 2, SEX 1, INC 1; and place an IDLE at "FTEST". The value of the constant at "CASREAD" is adjusted so that a consistent synchronization is realized. I can vary this constant between eight and ten and get consistent synchronization. The constant at SDELAY is chosen to position the samples for a particular value of the synchronization constant. Note 3 in the listing (at location 5D) refers to the fact that the PLO 5 is used to zero the "change in EF2" state counter. It is not needed for a functioning program, and has been included to allow the user to see the number of remaining transitions in the leading space.

The memory to cassette routine is somewhat intricate, especially in the treatment of the leader generation and initialization. Note 1 (at location 4) warns the user that DF must be set on entering "BT1" to generate a leader (or trailer). This routine starts output with ninety (hex) so that a left shift will set DF. To generate data starting at zero, we could use the instructions LDI X'80', SL, PLO 1.

Several miscellaneous remarks:

One bit cycle = 745.73868 machine cycles.
One 1200 Hz cycle = 186.43467 cycles.
One 2400 Hz cycle = 93.217335 machine cycles.

The listing provided is a list of a "hand-coded" assembly, (we don't have an assembler yet), and I hope there are no transcription errors.

Please let me know what you think.

```
00 90     CW17      GHI  0        SET LEADER STATUS - ZERO IF LEADER
01 A8               PLO  8          R8 CONTAINS LEADER STATUS
02 F890             LDI  90       GET FIRST ADDRESS TO BE OUTPUT
04 A1               PLO  1         (SEE NOTE 1 ABOUT THESE 3 INSTRUCTIONS)
05 FE               SL            SET DF TO GENERATE MARKS FOR LEADER
06 3047   CW17A     BR   BT1      GO SET UP ADDRESSES FOR DELAY CONSTANTS
08 06     MORETONE  LDN  6        GET PREVIOUS DELAY CONSTANT
09 FD08             SDI  X'08'    SET D IF PREVIOUS WAS A MARK CYCLE
0B 3006             BR   CW17A    WASTE TWO CYCLES GETTING TO BT1
0D A6     WLOOP     PLO  6        DELAY CONSTANT ADDRESS IS IN D
0E E1               SEX  1        WASTE TWO CYCLES
0F 7B               SEQ           UP PART OF TONE CYCLE
10 16               INC  6        MOVE ADDRESS POINTER BACK TO HIGH PART CONSTANT
11 C4               NOP           WASTE 3 CYCLES
12 06     WLA       LDN  6        LOAD DELAY CONSTANT
13 3225             BZ   WLB      ZERO IF LOW PART OF MARK
15 22               DEC  2        DECREMENT 2400 HZ COUNTER
16 FF01             SMI  X'01'     DELAY
18 3A16             BNZ  *-2         LOOP
1A 3925             BNQ  WLB       BRANCH IF LOW PART OF 1200 HZ CYCLE
1C 7A               REQ
1D 26               DEC  6        SET ADDRESS OF TONE CONSTANT TO LOW HALF OF TONE
1E 3012             BR   WLA      GO BACK FOR LOW PART OF TONE
           *        -             ADDRESS 20H IS USED BY THE QUEST MONITOR
                    ORG  X'21'
21 00     MARKL     DC   X'00'    MARK LOW  =       0 + 46 = 46
22 07     MARKH     DC   X'07'    MARK HIGH = 4*7 + 4 + 15 = 47
23 0B     SPACEL    DC   X'0B'    SPACE LOW = 4*11+ 4 + 46 = 94
24 12     SPACEH    DC   X'12'    SPACE HIGH= 4*18+ 4 + 15 = 91
25 88     WLB       GLO  8        GET LEADER STATE
26 3A2E             BNZ  WL1      NON-ZERO IF NOT LEADER
28 80               GLO  0        GET A NON-ZERO
29 3F00             BN4  CW17     MORE LEADER IF NO EF4
2B A8               PLO  8        SET LEADER STATUS TO NOT LEADER
2C 3034             BR   WL2      GO GENERATE SPACE IN FRONT OF FIRST DATA BITS
2E 82     WL1       GLO  2        GET 2400 CYCLE COUNTER
2F 3A3E             BNZ  BITTEST  ZERO MEANS END OF LAST STOP MARK FOR THIS BYTE
31 81               GLO  1        GET ADDRESS OF NEXT BYTE TO OUTPUT
32 FC00             ADDI X'00'    CONSTANT FOR LAST BYTE TEST
34 3200   WL2       BZ   CW17     GENERATE TRAILER IF DONE
36 01               LDN  1        DATA BYTE TO D
37 A4               PLO  4        DATA BYTE TO R4
38 F858             LDI  X'58'    (1SPACE + 8DATA BITS + 2MARKS)*8 = 88 = 58 HEX
3A A2               PLO  2        INITIALIZE 2400 HZ CYCLE COUNTER
3B 64               OUT  4        DISPLAY DATA BYTE AND INCREMENT R1
3C 3048             BR   BT2      GO GENERATE SPACE IN FRONT OF DATA BITS
3E FA07   BITTEST   ANDI X'07'    IF 2400 CYCLE COUNTER = 0(MOD 8) GET NEXT BIT
40 3A08             BNZ  MORETONE MORE TONE CYCLES FOR THIS BIT IF D NOT 0(MOD 8)
42 84               GLO  4        GET REMAINING DATA BITS
43 F6               SR            PUT BIT TO BE OUTPUT IN DF
44 F980             ORI  X'80'    FILL ON THE LEFT WITH ONES FOR TRAILING MARKS
```

```
46 A4                PLO  4         PUT REMAINING BITS BACK IN R4
47 C7      BT1       LSNQ
48 F821              LDI  MARKL     LOAD D WITH MARK HIGH CONSTANT ADDRESS IF DF=1
4A CF                LSQ
4B F823    BT2       LDI  SPACEL    LOAD D WITH SPACE HIGH CONSTANT ADDRESS IF DF=0
4D E1                SEX  1
4E 300D              BR   WLOOP     GO GENERATE HIGH PART OF TONE
50 F890    CR17      LDI  X'90'     FIRST ADDRESS TO BE LOADED TO R1
52 A1                PLO  1
53 5808    CASREAD   LDI  X'08'     LOOK FOR UP CYCLE 64 CYCLES LONG (SEE NOTE 2)
55 3D53    SYNC      BN2  CASREAD    8 CYCLE LOOP LOOKING
57 FF01              SMI  X'01'        AT EF2 AS MUCH
59 3D53              BN2  CASREAD        AS POSSIBLE
5B 3A55              BNZ  SYNC
5D A5                PLO  5         ZERO LEADING SPACE STATE COUNTER  (SEE NOTE 3)
5E 51                STN  1         ZERO BYTE TO BE LOADED
5F F809    NOBITS    LDI  X'09'     SET UP FOR THE START SPACE AND 8 DATA BITS
61 A4                PLO  4          (R4 IS THE BIT COUNTER)
62 F831    SDELAY    LDI  X'31'     (SEE NOTE 2)
64 3081    SA3       BR   SB2       START EF2 STATE SAMPLE
66 E1      STOREBIT  SEX  1
67 85                GLO  5         GET NUMBER OF TRANSITIONS IN EF2
68 FCF4    SETD      ADDI X'F4'     SET DF IF MORE THAN 11 TRANSITIONS
6A F0                LDX            GET PARTIAL BYTE
6B 76                SRC            SHIFT IN DF FROM THE LEFT
6C 51                STN  1         PUT BACK PARTIAL BYTE
6D 24                DEC  4         DECREMENT
6E 84                GLO  4           AND TEST BIT COUNTER
6F 3289              BZ   READOUT
71 90      SAMPLE    GHI  0         ZERO EF2 TRANSITIONS COUNTER
72 A5                PLO  5
73 F85A              LDI  X'5A'     90*8 + 26 = 746
75 3D64    SA1       BN2  SA3       HAS STATE CHANGED HIGH TO LOW?
77 15                INC  5          YES - INCREMENT R5
78 FF01    SA2       SMI  X'01'     CHECK FOR TIMEOUT
7A 3A7E              BNZ  SB1        AND GO LOOK FOR LOW TO HIGH TRANSITION
7C 3066              BR   STOREBIT
7E 3587    SB1       B2   SB3       HAS STATE CHANGED LOW TO HIGH?
80 15                INC  5          YES - INCREMENT R5
81 FF01    SB2       SMI  X'01'     CHECK FOR TIMEOUT
83 3A75              BNZ  SA1        AND GO LOOK FOR HIGH TO LOW TRANSITION
85 3066              BR   STOREBIT
87 3078    SB3       BR   SA2
89 64      READOUT   OUT  4         OUTPUT BYTE JUST READ
8A 81      FTEST     GLO  1         GET ADDRESS OF NEXT BYTE TO BE LOADED
8B FF00              SMI  X'00'     CONSTANT TO CHECK FOR LAST BYTE
8D 3A53              BNZ  CASREAD
8F 00                IDLE           IDLE IF DONE
                     END
```

```
                .TITLE  TICKET - WINNING TICKET DRAW
;
;                                T I C K E T
;
;                        WINNING TICKET DRAW PROGRAM
;
;
;   THIS PROGRAM IS USED TO RANDOMLY SELECT A NUMBER FROM A GIVEN
;   RANGE OF NUMBERS AND DISPLAY IT ON AN 8 DIGIT DISPLAY.  REFER
;   TO THE ARTICLE  "A LOW COST 8 DIGIT DISPLAY" BY BLAIR GERRISH
;   IN ISSUE NO. 5 OF IPSO FACTO , PAGE 49 , FOR A DESCRIPTION OF
;   THE DISPLAY HARDWARE.
;
;   PROGRAM USEAGE
;
;      1) THE PROGRAM SHOULD BE LOADED INTO MEMORY, STARTING AT
;         LOCATION #0000  ( NOTE # INDICATES A HEX NUMBER )
;      2) THE FIRST NUMBER OF THE SERIES SHOULD BE STORED, START-
;         ING AT MEMORY LOCATION 'FRSTNM' (#0002), 1 DIGIT PER
;         BYTE, MOST SIGNIFICANT DIGIT FIRST. THE NUMBER MAY BE UP
;         TO 8 DIGITS. THE PROGRAM HAS BEEN USED WITH ROLLS OF
;         TICKETS WHICH CAN BE PURCHASED QUITE CHEAPLY. THESE
;         TICKETS USUALLY CONTAIN 6 DIGIT NUMBERS.  IN THIS CASE
;         THE FIRST 2 BYTES (DIGITS) OF 'FRSTNM' WILL CONTAIN #00
;         EX.  ASSUME THAT THE FIRST TICKET NUMBER IS 400123.  TO
;              ENTER THE NUMBER YOU MUST LOAD THE  8  DATA  BYTES
;              INTO MEMORY STARTING AT ADDRESS #0002 BY KEYING IN
;               00, 00, 04, 00, 00, 01, 02, 03
;      3) THE LAST TICKET NUMBER +1 ( IE. THE NUMBER ON THE FIRST
;         TICKET LEFT ON THE ROLL )  SHOULD  NOW  BE  ENTERED  AT
;         ADDRESS 'LASTNM' (#000A).
;         EX.  ASSUME THAT 20 TICKETS WERE HANDED OUT SO THAT THE
;              FIRST  TICKET  LEFT  ON  THE ROLL IS 400143. ENTER
;              THE NUMBER AS 00, 00, 04, 00, 00, 01, 04, 03 .
;      4) RUN THE PROGRAM.  THE PROGRAM WILL CYCLE THROUGH THE
;         SEQUENCE OF NUMBERS, DISPLAYING EACH NUMBER FOR ABOUT 1
;         SECOND (1 MHZ CLOCK). THIS ALLOWS EACH TICKET HOLDER TO
;         SEE HIS/HER NUMBER AND BE REASSURED THAT  THE  COMPUTER
;         IS NOT CHEATING.
;      5) PRESS THE I-KEY. THE PROGRAM CYCLES THROUGH THE SEQUENCE
;         OF  NUMBERS  QUITE  QUICKLY.  WHEN  THE PROGRAM IS FULLY
;         TESTED THE SEP R9 INSTRUCTION AT LOCATION  #0041  SHOULD
;         BE  REMOVED  ( IE. CHANGED TO A NOP INSTRUCTION ) SO THAT
;         YOU CANNOT SEE THE NUMBERS BEING  INCREMENTED.  AT  THIS
;         TIME SOMEONE SHOULD BE SELECTED TO PICK A WINNING NUMBER.
;         ITS A GOOD IDEA TO STALL FOR ABOUT A MINUTE TO ALLOW THE
;         PROGRAM  TO  CYCLE THROUGH THE NUMBERS SEVERAL TIMES AND
;         THEREFORE ASSURE A MORE 'RANDOM' NUMBER.
;      6) PRESS THE I-KEY. A NUMBER WILL BE DISPLAYED. THIS IS YOUR
;         WINNING TICKET NUMBER.
;
                .SLW
```



```
                                                    1
                                                    2
                                                    3
                                                    4
                                                    5
                                                    6
                                                    7
                                                    8
                                                    9
                                                   10
                                                   11
                                                   12
0000  30 12                                        13
                                                   14
                                                   15
                                                   16
0002  =  00 08                                     17
000A  =  00 08                                     18
                                                   19
                                                   20
                                                   21
0012                                               22
0012  F8 49                                        23
0014  A8                                           24
0015  F8 9C                                        25
0017  A9                                           26
0018  F8 62                                        27
001A  AB                                           28
001B  F8 25                                        29
001D  A3                                           30
001E  F8 00                                        31
0020  B8                                           32
0021  B9                                           33
0022  BB                                           34
0023  B3                                           35
0024  D3                                           36
0025                                               37
0025  D8                                           38
0026                                               39
0026  F8 FF                                        40
0028  A4                                           41
0029                                               42
0029  D9                                           43
002A  24                                           44
002B  84                                           45
002C  3A 29                                        46
002E  DB                                           47
002F  3F 26                                        48
                                                   49
                                                   50
```

```
;
;   REGISTER USEAGE
;
;       R3  - P-COUNTER FOR THE MAIN PROGRAM
;       R4  - DELAY LOOP COUNTER FOR SLOW NUMBER DISPLAY
;       R8  - P-CNTR. FOR COPYNM RTN. ( COPY FRSTNM TO DSPNUM )
;       R9  - P-CNTR. FOR CHROUT RTN. ( DISPLAY NUMBER IN DSPNUM )
;       R11 - P-CNTR. FOR INCNUM RTN. ( INCREMENT NUMBER IN DSPNUM
;             AND COMPARE TO NUMBER IN LASTNM )
;       R13, R14, R15 - TEMPORARIES
;
;
        BR      START           ; SKIP AROUND DATA STOREAGE
;   STORE TICKET NUMBERS HERE
FRSTNM: .BLOCK  8               ; FIRST NUMBER IN SERIES
LASTNM: .BLOCK  8               ; LAST NUMBER IN SERIES + 1
;   INITIALIZATION SECTION
START:
        LDI     COPYNM          ; SUBR. TO COPY FRSTNM TO DSPNUM
        PLO     R8
        LDI     CHROUT          ; SUBR. TO DISPLAY NUMBER
        PLO     R9
        LDI     INCNUM          ; SUBR. TO INCREMENT DSPNUM
        PLO     R11             ;   AND COMPARE TO LASTNM
        LDI     INIT            ; MAIN PROGRAM P-COUNTER
        PLO     R3
        LDI     0
        PHI     R8              ; ZERO
        PHI     R9              ;  OUT HIGH
        PHI     R11             ;    BYTES OF
        PHI     R3              ;     REGISTERS
        SEP     R3
INIT:
        SEP     R8              ; INITIALIZE DISPLAYED NUMBER
SLOLUP:
        LDI     #FF             ; INITIALIZE DELAY LOOP COUNTER
        PLO     R4
DISPLY:
        SEP     R9              ; DISPLAY NUMBER
        DEC     R4              ; FOR A WHILE
        GLO     R4              ;
        BNZ     DISPLY
        SEP     R11             ; INCREMENT NUMBER
        BN4     SLOLUP          ; IF I NOT PRESSED DO NEXT
;
        .SLW
```

```
0031  F8 05          LDI     5               ; I PRESSED SO SET UP DELAY
0033  B4              PHI     R4              ;  FOR SOFTWARE DEBOUNCE
0034                  DEBNC1:
0034  24              DEC     R4
0035  94              GHI     R4
0036  3A 34          BNZ     DEBNC1
0038  37 38          B4      @
003A  F8 05          LDI     5
003C  B4              PHI     R4
003D                  DEBNC2:
003D  24              DEC     R4
003E  94              GHI     R4
003F  3A 3D          BNZ     DEBNC2
0041                  FASTLP:
0041  D9              SEP     R9              ; DISPLAY NUMBER
0042  D8              SEP     R11             ; INCREMENT NUMBER
0043  3F 41          BN4     FASTLP          ; REPEAT TILL I PRESSED
0045                  WINNER:
0045  D9              SEP     R9              ; DISPLAY WINNING NUMBER
0046  30 45          BR      WINNER          ;  FOR EVER AND EVER
                      ;
                      ;                       C O P Y N M
                      ;
                      ;    ROUTINE TO COPY FIRST NUMBER OF SERIES TO DISPLAY
                      ;    BUFFER ON INITIALIZE AND WHEN NUMBER INCREMENTED
                      ;    UP TO LAST NUMBER OF SERIES ( IE. RESET )
                      ;
0048  D3              COPYRT: SEP     R3              ; RETURN TO MAINLINE
0049  F8 02 AD       COPYNM: +DLDI   FRSTNM,R13      ; ADDRESS OF FIRST
004C  F8 00 BD
004F  F8 D8 AE       +DLDI   DSPNUM,R14      ; ADDRESS OF DISPLAY BUFFER
0052  F8 00 BE
0055  F8 08          LDI     8               ; INITIALIZE THE
0057  AF              PLO     R15             ;  DIGIT COUNTER
0058                  COPYLP:
0058  4D              LDA     R13             ; GET A DIGIT FROM FRSTNM
0059  5E              STR     R14             ; PUT IT IN DSPNUM
005A  1E              INC     R14             ;
005B  2F              DEC     R15             ; DECREMENT DIGIT COUNT
005C  8F              GLO     R15             ;  AND WHEN IT GETS TO
005D  3A 58          BNZ     COPYLP          ;   0 THEN FINISHED- NO
005F  30 48          BR      COPYRT          ; FINISHED SO RETURN

                      .SLN
```

```
                      ;                       I N C N U M
                      ;    ROUTINE TO INCREMENT THE NUMBER IN DSPNUM AND COMPARE
                      ;    IT TO THE VALUE IN LASTNM. IF DSPNUM IS GREATER THAN
                      ;    LASTNM THEN FRSTNM IS COPIED INTO DSPNUM. IE. INCREMENT
                      ;    AND RESET TO FIRST IF GREATER.
0061  D3              INCRTN: SEP     R3              ; R11= P COUNTER
0062  F8 DF AF       INCNUM: +DLDI   DSPNUM+7,R15    ; R15 POINTS TO LEAST
0065  F8 00 BF
                                                      ;  SIGNIFICANT DIGIT
0068  F8 08          LDI     8               ; R14 IS A DIGIT COUNTER
006A  AE              PLO     R14
006B                  INCLUP:
006B  0F              LDN     R15             ; GET A DIGIT (LEAST SIGN.
006C  FD 09          SDI     9               ;  FIRST) AND COMPARE TO 9
006E  3A 78          BNZ     INCDIG          ; IF DIGIT < 9 THEN INC.
0070  5F              STR     R15             ;  ELSE SET TO ZERO
0071  2E              DEC     R14             ; DECREMENT DIGIT COUNT AND
0072  8E              GLO     R14             ;  CHECK FOR ALL DIGITS DONE
0073  32 7C          BZ      CMPNUM          ;   AND IF SO THEN COMPARE
0075  2F              DEC     R15             ;    ELSE DECR. TO NEXT DIGIT
0076  30 6B          BR      INCLUP          ;     AND REPEAT TEST
0078                  INCDIG:
0078  0F              LDN     R15             ; INCREMENT A DIGIT IF
0079  FC 01          ADI     1               ;  PREVIOUS DIGIT WAS A 9
007B  5F              STR     R15
007C                  CMPNUM:
007C  F8 0A AF       +DLDI   LASTNM,R15      ; R15 POINTS TO LAST NUMBER
007F  F8 00 BF
0082  F8 D8 AE       +DLDI   DSPNUM,R14      ; R14 POINTS TO WORKING STORAGE
0085  F8 00 BE
0088  EE              SEX     R14
0089  F8 08          LDI     8               ; R13 IS A DIGIT COUNTER
008B  AD              PLO     R13
008C                  CMPLUP:
008C  0F              LDN     R15             ; GET A DIGIT FROM ENDING NUMBE
008D  F5              SD                      ;  SUB. A DIGIT FROM WORK AREA
008E  3A 61          BNZ     INCRTN          ;   IF NOT EQUAL THEN RETURN
0090  2D              DEC     R13             ; ELSE DECREMENT DIGIT COUNT
0091  8D              GLO     R13             ;  AND CHECK FOR ALL DIGITS DON
0092  32 98          BZ      CMPEQL          ;   IF DONE THE NUMBERS EQUAL
0094  1E              INC     R14             ;   IF NOT DONE THEN INC.
0095  1F              INC     R15             ;    TO NEXT DIGITS AND
0096  30 8C          BR      CMPLUP          ;     REPEAT THE TESTS
0098                  CMPEQL:
0098  D8              SEP     R8              ; COPY FRSTNM TO DSPNUM
0099  30 62          BR      INCNUM

                      .SLN
```

```
                                ;                    C H R O U T
                                ;
                                ;              CHARACTER OUTPUT ROUTINE
                                ;
009B  D3            CHRRTN: SEP     R3              ; RETURN TO MAIN PROGRAM
009C  F8 D8 AF      CHROUT: +DLDI   DSPNUM,R15      ; R15 POINTS TO DATA TABLE
009F  F8 00 BE
00A2  F8 01                 LDI     1
00A4  BC                    PHI     R12             ; R12.HI IS DIGIT SELECT
00A5  ED                    SEX     R13
00A6  F8 C7 AD      CHRLUP: +DLDI   DIGSEL,R13      ; R13 = END OF OUTPUT TABLE
00A9  F8 00 BD
00AC  9C                    GHI     R12             ; GET DIGIT SELECT BIT AND SAVE
00AD  73                    STXD    R13             ;   THEN DEC. X TO POINT TO SEGSEL
00AE  FE                    SHL                     ; SHIFT DIGIT SELECT BIT FOR
00AF  BC                    PHI     R12             ;   NEXT DISPLAY DIGIT AND SAVE
00B0  4F                    LDA     R15             ; MASK OFF ANY UNUSED DATA BITS
00B1  FA 0F                 ANI     #0F             ;   JUST IN CASE THEY ARE SET
00B3  FC C8                 ADI     CHRTAB          ; ADD LOW ADDR. OF SEGMENT
00B5  AE                    PLO     R14             ;  SELECT TABLE AND SAVE
00B6  F8 00                 LDI     CHRTAB_-8       ; GET HIGH ADDR. OF SEG. SEL. TAB.
00B8  7C 00                 ADCI    0               ;  AND ADD IN CARRY FROM PREVIOUS
00BA  BE                    PHI     R14             ;   THEN SAVE HIGH ADDRESS
00BB  0E                    LDN     R14             ; GET SEGMENT SELECT DATA AND
00BC  73                    STXD                    ;  PUT IN OUTPUT TABLE
00BD  62                    OUT     2               ; OUTPUT DIGIT DESELECT
00BE  61                    OUT     1               ; OUTPUT SEGMENT SELECT
00BF  62                    OUT     2               ; OUTPUT DIGIT SELECT
00C0  9C                    GHI     R12             ;  AND CHECK FOR MORE TO DO
00C1  3A A6                 BNZ     CHRLUP          ; LOOP FOR NEXT IF NOT ALL DONE
00C3  30 9B                 BR      CHRRTN          ; RETURN IF ALL DONE

00C5  00            DIGBLK: .BYTE   0               ; DIGIT BLANK CODE
00C6  00            SEGSEL: .BYTE   0               ; SEGMENT SELECT CODE
00C7  00            DIGSEL: .BYTE   0               ; DIGIT SELECT BIT

                    ; TABLE TO SELECT CORRECT SEGMENTS TO BE DISPLAYED
                    ;       NOTE SET BIT = SEGMENT ON

00C8                CHRTAB:
00C8  3F                    .BYTE   #3F     ; 0
00C9  30                    .BYTE   #30     ; 1
00CA  5B                    .BYTE   #5B     ; 2
00CB  4F                    .BYTE   #4F     ; 3
00CC  66                    .BYTE   #66     ; 4
00CD  6D                    .BYTE   #6D     ; 5
00CE  7D                    .BYTE   #7D     ; 6
00CF  07                    .BYTE   #07     ; 7
00D0  7F                    .BYTE   #7F     ; 8
00D1  67                    .BYTE   #67     ; 9
00D2  77                    .BYTE   #77     ; A
00D3  7C                    .BYTE   #7C     ; B (SMALL)
00D4  39                    .BYTE   #39     ; C
00D5  5E                    .BYTE   #5E     ; D (SMALL)
00D6  79                    .BYTE   #79     ; E
00D7  71                    .BYTE   #71     ; F

00D8  =  00 08      DSPNUM: .BLOCK  8       ; NUMBER DISPLAY ETC. BUFFER
00D8  =  00 12              .END    START
```

CKET - WINNING TICKET DRAW RC1802-V01 TICKET.RCA 15-JUN-78

The Association of Computer Experimenters

Minutes of Club Meeting 78-4

Held at Stelco Wilcox St. Auditorium

6 April, 1978 8:00 P.M.

78-4-1 The regular meeting was preceded by a 1 hour tutorial.

78-4-2 Motion to adopt Minutes 78-3
Proposed - George York
Seconded - Sy Cobb
Carried unanimously

78-4-3 George York, secretary-treasurer, reported a current bank balance of $1381.32.

Re 78-3-3 - information from the Ministry of Consumer and Commercial Relations indicates the club cannot incorporate for at least one year. Therefore, any move in this regard must be done by a future executive.

A call was made for nominations for the new club executive.

78-4-4 Membership. There are now 379 paid members.

78-4-5 Software. Bernie Murphy noted the importance of setting a standard format for cassette interface before the summer.

78-4-6 Hardware. No report.

78-4-7 Ken Bevis has suggested setting up a phone contact committee. This would allow information exchange without long-distance phone tolls. Bernie Murphy volunteered as an Oakville/ Mississauga contact and Bert deKat volunteered as a Brantford/ Linden region contact.

78-4-8 Brian Fox is attempting to set up a ham radio net. There are a large number of 1802 users that are also hams. It was noted that Ed Leslie can contact any ham in North America.

78-4-9 Other Business.

(a) Stephen Fielding of Hamilton Logic Systems, P.O. Box 7, Stoney Creek is setting up a mail order distribution centre for electronics parts.

(b) Ken Christmas mentioned the club should be thinking about the size of the membership and the size of the newsletter. He noted that we may be growing out of control

(c) A standard format for articles submitted to the newsletter should be detailed in the newsletter. This would make the newsletter easier to assemble.

78-4-9 Other Business (cont'd)

(d) Door Prize draw for Beldon ribbon cable. The winners were: John Rogers
Walt Smith
Joe Flaherty

(e) A number of desk type junk calculators (good for parts) were donated to the club. These were made available for pick-up by members attending the meeting as a "bonus door prize".

78-4-10 Ken Collins gave a very interesting talk on the theory of wire-wrapping and followed the talk with a demonstration of various equipment and techniques.

78-4-11 Motion to adjourn meeting
Proposed - Tom Crawford
Seconded - Wayne Bowdish
Passed

The meeting adjourned at 10:00 P.M.
About 40 people attended the meeting.

The Association of Computer Experimenters

Minutes of Club Meeting 78-5

Held at Stelco Wilcox St. Auditorium

9 May, 1978 8:00 P.M.

78-5-1 There was some discussion about board by Data 100 featuring an 1802 CPU designed for an S100 bus. Some interest was expressed.

78-5-2 There are presently 434 paid members. The bank balance is $1137.62.

78-5-3 38 people have replied to Lyle Sandy's system file report.

78-5-4 It was reported that few people had sent in hardware/applications. Most people sending in info or articles are computer/large systems oriented.

78-5-5 There was a discussion and demonstration of a Video Display and ASCII I/O in general.

78-5-6 Door prize draws were held for a number of pieces of ribbon cable.

78-5-7 Bernie Murphy outlined his proposal for a cassette software data format standard for the club. The proposal will be published in the next newsletter.

Motion - that the ACE Club formally adopt the standard presented in the document called "Software Standard for Kansas City Tape" that will appear in the next newsletter.
Proposed - Bernie Murphy
Seconded - Vic Sydiuk
Passed

78-5-8 Motion to adjourn meeting
Proposed - Tom Crawford
Seconded -
Passed

The Association of Computer Experimenters

Minutes of Annual General Club Meeting 78-6

Held at Stelco Wilcox St. Auditorium

23 May, 1978 8:00 P.M.

78-6-1 George York, secretary-treasurer, reported a current membership of 450 paid members and a current bank balance of $1274.48.

78-6-2 Motion to adopt Minutes 78-4
Proposed - George York
Seconded - Tom Crawford
Carried unanimously

78-6-3 Ken Smith mentioned there was a small but solid core of people interested in the training sessions.

78-6-4 Hardware report by Eugene Tekatch mentioned that the club had progressed exceptionally well considering the lack of 1802 software support--the club has developed as a grassroots organization. However, the club is lacking in hardware and varied applications expertise. Eugene outlined hardware developments of Tektron and pointed out that now the memory and some peripheral hardware is available, the emphasis of the hardware co-ordination should be on applications.

78-6-5 Constitutional Amendment--as published in Newsletter #5, page 56, section 78-2-9.
Motion to adopt the Constitutional Amendment as published
Proposed - Tom Crawford
Seconded - Geff Waite
Carried unanimously

Article 5.4 of the Constitution now reads:
Meetings of the executive committee shall be at the call of the president or, in his absence, the secretary/treasurer.

78-6-6 Club T-shirts were discussed. George York is to enquire of a local manufacturer and write a note for the next newsletter. A "Club Logo" contest will be tried-the winner to get 2 free T-shirts.

78-6-7 Audit Committee volunteers were requested. Vic Sydiuk and Mike Hrycko volunteered. The club books will be audited as soon as the current year's expenses can be finalized.

78-6-8 Election of 1978-79 Executive. The office of President was the only contested position. Norm Cunningham and Ken Bevis were nominees--election by secret ballot resulted

78-6-8 (cont'd)
Ken Bevis elected as the new President. The other positions were declared by acclamation. The list of new executive will be published in Issue #6.

78-6-9 Motion to adjourn meeting
Proposed - Steve Fielding
Seconded - Mike Hrycko
Passed

The meeting adjourned at 9:40 P.M.
20 members attended the meeting.

ACE CLUB T-SHIRTS & LOGO CONTEST

At the Annual General Meeting the subject of club T-shirts was discussed. Eugene Tekatch has volunteered to supply the silk screen for the T-shirts and a local manufacturer can supply the T-shirts for a cost of about $4.00 each in quantities of 36 or more.

One suggestion for a design is a pin-out diagram of an 1802 chip. In addition, or in place of the pin-out, a club logo could be used.

ATTENTION !!!!!
ALL CLUB MEMBERS TAKE NOTE

CLUB LOGO DESIGN CONTEST

Designs for a club logo should be sent to the new President, Ken Bevis. Entries should be on 8½ x 11 plain paper.

GRAND PRIZE - TWO FREE T-SHIRTS

Latest in the rumour market - the IEEE Hamilton Section is tentatively planning to have an Advanced Microprocessor/ Applications course in the coming year. This would possibly be a follow-on to the initial course held last year.

NOTICE OF CLUB MEMBERSHIP RENEWAL & NEW CLUB DUES ASSESSMENT

All club memberships expire 31 May, 1978. Club members should complete the form below and mail to George York, 60 Chester Road, Stoney Creek, Ontario, Canada, L8E 1Y2.

The club dues assessment for the 1978-79 fiscal year is $10.00. Make cheques payable to the Association of Computer Experimenters.

If you know of anyone who would be interested in joining our club, why not give him/her the membership application. We are constantly looking for new members with new and interesting ideas.

MEMBERSHIP APPLICATION FORM FOR THE Association of Computer Experimenter

FIRST NAME

LAST NAME

FIRST LINE OF ADDRESS

SECOND LINE OF ADDRESS

CITY & PROVINCE

POSTAL CODE

Association of Computer Experimenters

MEMBERSHIP RENEWAL FORM

ATTACH MAILING LABLE HERE

FIRST NAME

LAST NAME

FIRST LINE OF ADDRESS

SECOND LINE OF ADDRESS

CITY & PROVINCE

POSTAL CODE